On The Delays In Spiking Neural P Systems

Francis George C. Cabarle, Kelvin C. Buño, Henry N. Adorna
Algorithms & Complexity Lab
Department of Computer Science
University of the Philippines Diliman
Diliman 1101 Quezon City, Philippines
{fccabarle, kcbuno, hnadorna}@up.edu.ph

ABSTRACT
In this work we extend and improve the results done in a previous work on simulating Spiking Neural P systems (SNP systems in short) with delays using SNP systems without delays. We simulate the former with the latter over sequential, iteration, join, and split routing. Our results provide constructions so that both systems halt at exactly the same time, start with only one spike, and produce the same number of spikes to the environment after halting.

Keywords: Membrane Computing, Spiking Neural P systems, delays, simulation, routing

1. INTRODUCTION
SNP systems, first presented in [7] and with some recent results in [12], [13] and [10] (among others), are computing devices inspired by how biological neurons represent information: using electro-chemical signals called spikes. Since spikes are indistinct, information is taken not from the spikes themselves, but from their multiplicity or time of arrival. One motivation for SNP systems (as is the case in the area of Membrane Computing [11] in general) is to abstract ideas from biology for computational use. For SNP systems in particular, the neuron from our brains is the motivation. It can be argued that the human brain is one (if not currently) the most complicated and powerful “supercomputer” known to us at the moment. The brain performs complex computations from billions of interconnected neurons while consuming only around 10 to 20 Watts of energy [8], and it is small enough to fit in our skulls. It is therefore desirable to work with as little quantity of “energy” as possible, and we can think of the spike in SNP systems as being such quantity.

SNP systems are Turing complete devices [7, 5] and have been used as (among others) transducers [6], generating vectors of numbers [1], as well solving hard problems [9]. Spiking rules (rules that produce spikes) are usually of two types: with delays and without delays. If an SNP system has at least one rule with a delay, we refer to it as an SNP system with delay labeled as Π, otherwise it is known as an SNP system without delay labeled as \( \overline{\Pi} \).

In this work we extend the work presented in [3], with the goal of simulating a Π that performs sequential, iteration, join, and split routing with a \( \overline{\Pi} \) that performs the same routings. By routing we mean the transfer or movement of spikes from one neuron to another. By simulation in [3] it is meant that the following two requirements are satisfied:

\[ R_1 : \text{Halting time of } \overline{\Pi} \text{ coincides with the halting time of } \Pi, \] or is offset either by a fixed timestep or by a function of the delays in \( \Pi \),

\[ R_2 : \text{number of spikes in the final configuration of } \Pi \text{ is the same number in } \overline{\Pi}, \] or is offset by a function of the delays in \( \Pi \).

In [3], the construction of \( \overline{\Pi} \) from \( \Pi \) is such that the initial spikes of \( \overline{\Pi} \) is a function of the delay (or delays) in \( \Pi \). In particular, the initial spikes of \( \overline{\Pi} \) are multiples of the delays in \( \Pi \). Aside from the increased initial spike number in \( \overline{\Pi} \), the exponents in the regular expressions and the number of consumed spikes of certain spiking rules in \( \overline{\Pi} \) are also multiples of all the delays in a given routing of \( \Pi \).

In this work we extend the work presented in [3], with the goal of simulating a Π that performs sequential, iteration, join, and split routing with a \( \overline{\Pi} \) that performs the same routings. By routing we mean the transfer or movement of spikes from one neuron to another. By simulation in [3] it is meant that the following two requirements are satisfied:

\[ R_1 : \text{Halting time of } \overline{\Pi} \text{ coincides with the halting time of } \Pi, \] or is offset either by a fixed timestep or by a function of the delays in \( \Pi \),

\[ R_2 : \text{number of spikes in the final configuration of } \Pi \text{ is the same number in } \overline{\Pi}, \] or is offset by a function of the delays in \( \Pi \).

We improve the work done in [3] by providing alternative constructions in this work. Our specific contributions are as follows:

- we construct a \( \overline{\Pi} \) that simulates a Π that performs sequential, iteration, join, and split routing,
- both \( \overline{\Pi} \) and Π start with only one spike each in the initial neuron,
- halting time of \( \overline{\Pi} \) and Π coincide i.e. there are no offsets,
number of spikes sent to the environment after halting are equal for \( \Pi \) and \( \overline{\Pi} \).

our construction allows split routing even if the delays of the output neurons are not equal.

The trade-off is that for every delay \( d \) in \( \Pi \), we add \( d \) neurons in \( \overline{\Pi} \). If the initial neuron of \( \Pi \) has a delay, following our construction means we simply modify \( \Pi \) and \( \overline{\Pi} \) such that their new halting time involves one additional time step. The succeeding sections are as follows: Section 2 provides preliminaries and assumptions for our work. Section 3 presents our main results. We end with our final remarks and directions for future work in Section 4.

2. PRELIMINARIES

It is assumed that the readers are familiar with the basics of Membrane Computing (a good introduction is [11] with recent results and information on the P systems webpage at http://ppage.psystems.eu/) and formal language theory. We only briefly mention notions and notations which will be useful throughout the paper.

Let \( V \) be an alphabet, \( V^* \) is the free monoid over \( V \) with respect to concatenation and the identity element \( \lambda \) (the empty string). The set of all non-empty strings over \( V \) is denoted as \( V^+ \) so \( V^+ = V^* - \{\lambda\} \). We call \( V \) a singleton if \( V = \{a\} \) and simply write \( a^* \) and \( a^+ \), instead of \( \{a^\} \) and \( \{a^+\} \). The length of a string \( w \in V^+ \) is denoted by \( |w| \). If \( a \) is a symbol in \( V \), \( a^\) = \( \lambda \). A language \( L \subseteq V^* \) is regular if there is a regular expression \( E \) over \( V \) such that \( (\Pi(E) = L) \). A regular expression over an alphabet \( V \) is constructed starting from \( \lambda \) and the symbols of \( V \) using the operations union, concatenation, and \( + \), using parentheses when necessary to specify the order of operations. Specifically, (i) \( \lambda \) and each \( a \in V \) are regular expressions, (ii) if \( E_1 \) and \( E_2 \) are regular expressions over \( V \), then \( (E_1 \cup E_2), (E_1 E_2), \) and \( E_1^+ \) are regular expressions over \( V \), and (iii) nothing else is a regular expression over \( V \). With each expression \( E \) we associate a language \( L(E) \) defined in the following way: (i) \( L(\lambda) = \{\lambda\} \) and \( L(a) = \{a\} \) for all \( a \in V \), (ii) \( L(E_1 \cup E_2) = L(E_1) \cup L(E_2) \), \( L(E_1 E_2) = L(E_1) L(E_2) \), and \( L(E_1^+) = L(E_1)^+ \), for all regular expressions \( E_1, E_2 \) over \( V \). Unnecessary parentheses are omitted when writing regular expressions, and \( E^+ \cup \{\lambda\} \) is written as \( E^* \). Next we have the definition for an SNP system.

Definition 1 (SNP system). An SNP system of a finite degree \( m \geq 1 \) is a construct of the form

\[
\Pi = (O, \sigma_1, \ldots, \sigma_m, \text{syn}, \text{out}),
\]

where:

1. \( O = \{a\} \) is the singleton alphabet (\( a \) is called spike).
2. \( \sigma_1, \ldots, \sigma_m \) are neurons of the form \( \sigma_i = (n_i, R_i) \), 1 \( \leq i \leq m \), where:
   
   (a) \( n_i \geq 0 \) is an integer representing the number of spikes in \( \sigma_i \),

(b) \( R_i \) is a finite set of rules of the general form

\[
E/a^c \rightarrow a^b; \ d
\]

where \( E \) is a regular expression over \( O \), \( c \geq 1 \), if \( b > 0 \) then \( d \geq 0 \) and \( c \geq b \), else if \( b = 0 \) then \( d = 0 \).

3. \( \text{syn} \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\} \), \( (i, i) \notin \text{syn} \) for \( 1 \leq i \leq m \), are synapses between neurons.

4. \( \text{out} \in \{1, 2, \ldots, m\} \) is the index of the output neuron.

A spiking rule is where \( b \geq 1 \). A forgetting rule is a rule where \( b = 0 \) is written as \( E/a^c \rightarrow \lambda \). If \( L(E) = \{a^c\} \) then spiking and forgetting rules are simply written as \( a^c \rightarrow a^b \) and \( a^c \rightarrow \lambda \), respectively. Applications of rules are as follows: if neuron \( \sigma_i \) contains \( k \) spikes, \( a^k \in L(E) \) and \( k \geq c \), then the rule \( E/a^c \rightarrow a^k \in R_i \) is enabled and the rule can be fired or applied. If \( b \geq 1 \), the application of this rule removes \( c \) spikes from \( \sigma_i \), so that only \( k - c \) spikes remain in \( \sigma_i \). The neuron sends \( b \) number of spikes to every \( \sigma_j \) such that \( (i, j) \in \text{syn} \). The output neuron has a synapse not directed to any other neuron, only to the environment. The neuron \( \sigma_i \) is referred to as the initial neuron.

If a spiking rule (forgetting rules cannot have delays) has \( d = 0 \), the \( b \) number of spikes are sent immediately i.e. in the same time step as the application of the rule. If \( d \geq 1 \) and the spiking rule was applied at time \( t \), then the spikes are sent at time \( t + d \). From time \( t \) to \( t + d - 1 \) the neuron is said to be closed (inspired by the refractory period of the neuron in biology) and cannot receive spikes. Any spikes sent to the neuron when the neuron is closed are lost or removed from the system. At time \( t + d \) the neuron becomes open and can then receive spikes again. The neuron can then apply another rule at time \( t + d + 1 \). If \( b = 0 \) then no spikes are produced. SNP systems assume a global clock, so the application of rules and the sending of spikes by neurons are all synchronized.

A configuration of the system at time \( k \) is denoted as \( C_k = (n_1/t_1, \ldots, n_m/t_m, n_e) \), where each element of the vector (except for \( n_e \), denoting the spikes in the environment) is the configuration of a neuron \( \sigma_i \), with \( n_i \) spikes and is open after \( t_i \geq 0 \) steps. An initial configuration \( C_0 \) is therefore \( (n_1/0, \ldots, n_m/0, 0) \) since no rules whether with or without delay, have yet been applied and the environment is initially empty. A computation is a sequence of transitions from an initial configuration. A computation may halt (no more rules can be applied for a given configuration) or not. If an SNP system does halt, all neurons should be open. Computation result in this work is obtained by checking the number of spikes in the environment once the system halts.

Figure 1: SNP system with delay \( \Pi_0 \).
As an example, let us have an SNP system shown in Figure 1 formally defined as follows: \( \Pi_0 = (O, \sigma_1, \sigma_2, \sigma_3, \text{syn, out}) \) where \( \sigma_1 = (1, a^+/a \rightarrow a) \), \( \sigma_2 = (0, a^+/a \rightarrow a; 2) \), \( \sigma_3 = (0, a^+/a \rightarrow a) \), \( \text{syn} = \{ (1,2), (2,3) \} \), initial neuron is \( \sigma_1 \), and \( \text{out} = 3 \). Only neuron \( \sigma_1 \) has one spike initially and only \( \sigma_2 \) has a rule with a delay \( d = 2 \). We have \( C_0 = (1/0, 0/0, 0/0, 0) \). At the next step, \( \sigma_1 \) can use its rule (it has at least one spike) and consumes one spike and sends one spike immediately to \( \sigma_2 \) so we have \( C_1 = (0/0, 1/0, 0/0, 0) \). At step 2, \( \sigma_2 \) consumes its spike and closes for 2 time steps, so \( C_2 = (0/0, 0/2, 0/0, 0) \). At step 3 we have \( C_3 = (0/0, 0/1, 0/0, 0) \). At time step 4, \( \sigma_2 \) opens and sends one spike to \( \sigma_3 \), so \( C_4 = (0/0, 0/0, 1/0, 0) \). Finally, at time step 5 the output neuron sends one spike to the environment, \( \Pi_0 \) halts and we have \( C_5 = (0/0, 0/0, 0/0, 1) \).

Notice that iteration routing can be formed by combining the three other constructs. Also notice that if there exists a sequential path from \( \sigma_i \) (with delay \( d_1 \)) to \( \sigma_j \) (with delay \( d_2 \)) so that \( d_1 < d_2 \) and the number of spikes of the initial neuron \( \sigma_1 \) in \( C_0 \) is \( n_1 > 1 \), it is possible for some spikes to be lost. The reason is that it is possible for \( \sigma_i \) to still be closed when spikes from \( \sigma_i \) arrive. We avoid lost spikes by considering SNP systems where the initial neuron has only one spike. We say in this work that a \( \Pi \) simulates a \( \Pi \) if two requirements are satisfied:

\( R_1 : \) halting time of \( \Pi \) spikes is the same halting time of \( \Pi \).

\( R_2 : \) number of spikes in the environment of \( \Pi \) when \( \Pi \) halts is equal to the number of spikes in the environment of \( \Pi \) when \( \Pi \) halts.

### 3. MAIN RESULTS

We begin presenting our results with a fundamental idea on sequential routing.

**Lemma 1 (Sequential routing).** Given an SNP system with delay \( \Pi \) performing sequential routing, there exists an SNP system without delay \( \Pi \) performing sequential routing that simulates \( \Pi \).

**Proof.** We refer to Figure 3 for illustrations. Let

\[ \Pi = (O, \sigma_{11}, \sigma_{12}, \{ (11, 12) \}, 12) \]  
and \( \sigma_2 = (0, a^+/a \rightarrow a; d) \)

we then let

\[ \Pi' = (O, \sigma_{21}, \sigma_{22}, \{ \text{syn, 23} \} ) \]  
where \( 1 \leq i \leq d \), \( \text{syn} = \{ (21, 22-1), \ldots, (21, 22-(d-1)), (22-1, 22-d), \ldots, (22-(d-1), 22-d), (22-d, 23) \} \) \( \sigma_{21} = (1, a^+/a \rightarrow a) \), \( \sigma_{22} = (0, a^+/a \rightarrow a) \), \( \sigma_{23} = (0, a^{d-1}+/a^{d-1} \rightarrow a) \).

The additional \( d \) neurons, immediately after initial neuron \( \sigma_{21} \) in \( \Pi' \), are used to multiply the single spike from \( \sigma_{21} \). The additional neurons then send one spike each to \( \sigma_{22, \ldots} \). Neuron \( \sigma_{22, \ldots} \) accumulates \( d - 1 \) spikes, and consumes these, one spike at a time and sending one spike every time to \( \sigma_{23} \). This consumption of one spike every time step creates a delay of \( d - 1 \) time steps. Due to the regular expression of the rule in \( \sigma_{23} \), the neuron will have to accumulate \( d - 1 \) spikes before the rule is used. Once \( \sigma_{23} \) accumulates \( d - 1 \) spikes, it immediately sends one spike to the environment. This spiking and halting occurs at time \( t + d + 1 \) for both \( \Pi \) and \( \Pi' \) (satisfying \( R_1' \) and \( R_2' \) if we let \( t \) be the time when \( \sigma_{11} \) and \( \sigma_{21} \) spike.

We can repeatedly apply the previous construction if there exist more than one neuron with a (rule having a) delay in a sequential path as seen in Figure 4. It can be easily shown that if there exists \( \sigma_i \) without delay in a sequential path between \( \sigma_{11} \) and \( \sigma_{21} \), the time to halt for both \( \Pi_0 \) and \( \Pi' \) still coincide. In particular, every additional \( \sigma_i \) having a rule without a delay adds one time step to the halting time of both \( \Pi \) and \( \Pi' \). Both \( R_1' \) and \( R_2' \) are still satisfied. \( \square \)
Π uses the construction idea in Lemma 1. In Figure 5 an infinite loop is created:

Figure 3: Sequential routing: \( \Pi_1 \) (top) with delay \( d \),  
and \( \Pi_1 \) (bottom) simulating \( \Pi_1 \).

A sample computation of \( \Pi_1 \) and \( \Pi_1 \) is shown in Table 1.  
For sample computations of \( \Pi_2 \) and \( \Pi_2 \) we refer to Table 2.  
From Lemma 1 we have the following observation.

**Observation 1.** If \( \Pi \) has more than one neuron with a delay in a rule, the total additional neurons in \( \Pi \) is \( \sum_i d_i \)  
where \( d_i \) is the delay of the rule in \( \sigma_i \).

| Steps | \( \Pi_1 \) | \( \Pi_1 \) |
|-------|-----------|-----------|
| \( t_0 \) | \( 1/0, 0/0, 0/0 \) | \( 1/0, 0/0, 0/0 \) |
| \( t_1 \) | \( 0/0, 1/0, 0/0 \) | \( 0/0, 1/0, 0/0 \) |
| \( t_2 \) | \( 0/0, 0/2, 0/0 \) | \( 0/0, 1/0, 0/0 \) |
| \( t_3 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |
| \( t_4 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |

Table 1: Sample computations of \( \Pi_1 \) and \( \Pi_1 \), \( d = 3 \).

| Steps | \( \Pi_2 \) | \( \Pi_2 \) |
|-------|-----------|-----------|
| \( t_0 \) | \( 1/0, 0/0, 0/0 \) | \( 1/0, 0/0, 0/0 \) |
| \( t_1 \) | \( 0/0, 1/0, 0/0 \) | \( 0/0, 1/0, 0/0 \) |
| \( t_2 \) | \( 0/0, 0/2, 0/0 \) | \( 0/0, 1/0, 0/0 \) |
| \( t_3 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |
| \( t_4 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |
| \( t_5 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |
| \( t_6 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |
| \( t_7 \) | \( 0/0, 0/0, 0/0 \) | \( 0/0, 0/0, 0/0 \) |

Table 2: Sample computations of \( \Pi_2 \) and \( \Pi_2 \), \( d_1 = 2 \),  
\( d_2 = 3 \).

**Lemma 2.** (Iteration routing). Given an SNP system with delay \( \Pi \) performing iteration routing, there exists an SNP system without delay \( \Pi \) performing iteration routing that simulates \( \Pi \).

**Proof.** We refer to Figure 5 for illustrations. Let

\[
\Pi = (O, \sigma_{11}, \sigma_{12}, \{11, 12\}, 12) \text{ where } \sigma_{11} = (1, a^+/a \rightarrow a; d), \sigma_{12} = (0, a^+/a \rightarrow a)
\]

we then let

\[
\Pi = (O, \sigma_{21}, \sigma_{22}, \{21, 22\}, 22) \text{ where } \sigma_{21} = (1, a^+/a \rightarrow a), \sigma_{22} = (0, a^+/a \rightarrow a) \text{ for } 1 \leq i \leq d, \sigma_{23} = (0, (a^{d-1})^+/a^{d-1} \rightarrow a) \text{ and } syn = \{(21, 22-1), \ldots, (21, 22-(d-1)), (22-1, 22-d), \ldots, (22-(d-1), 22-d), (22-d, 23), (23, 21)\}, \text{ out = 23}
\]

The construction of \( \Pi \) uses the construction idea in Lemma 1. i.e. the neuron with a delay \( d \) in \( \Pi \) is replaced with \( d \) additional neurons in \( \Pi \). In Figure 6 an infinite loop is created: a spike starts at \( \sigma_{11} \) and it uses its rule at time \( t \) so that the spike is sent to \( \sigma_{12} \) at time \( t + d \), then \( \sigma_{12} \) immediately sends a spike back to \( \sigma_{11} \) (and the environment) at time \( t + d + 1 \), and so on and so forth. Similarly, \( \sigma_{21} \) sends a spike to neurons \( \sigma_{22-1} \) to \( \sigma_{22-d-1} \) at time \( t \). At time \( t + 1 \), \( \sigma_{22-d} \) accumulates \( d - 1 \) spikes from the \( d - 1 \) neurons from the previous time step. The spikes in \( \sigma_{22-d} \) are consumed and then sent one at a time to \( \sigma_{22-d} \). At time \( t + d \), \( \sigma_{22-d} \) accumulates \( d - 1 \) spikes so that it sends one spike back to \( \sigma_{21} \) and at the environment at time \( t + d + 1 \), coinciding with the time of spiking of \( \sigma_{12} \). Thus, \( R_1 \) and \( R_2 \) are satisfied.

**Lemma 2** for iteration routing makes use of the construction used in Lemma 1 for sequential routing. This construction will again be used for the join and split routings as follows.

**Observation 2.** If the initial neuron of \( \Pi \) has a delay and its halting time is \( t + d \), we add a new initial neuron \( \sigma_{11} \) in \( \Pi \) with \( 1^* \) in \( \Pi \) so that \( \Pi \) halts at time \( t + d + 1 \). We then add a new initial neuron similarly to \( \Pi \) and modify its syn (following Lemma 1 construction) so that \( \Pi \) halts at \( t + d + 1 \) instead, simulating \( \Pi \).

Although the premise of Observation 2 is different from our assumption in Section 2 that the initial neuron has no delay, the observation provides a solution on how to approach such a premise. For example, if \( \sigma_{11} \) has a delay instead of \( \sigma_{23} \) in \( \Pi_3 \), we add a new initial neuron to \( \sigma_{12} \) and a
new synapse \((11', 11) \in \text{syn}\) in \(\Pi\). For \(\Pi_3\), we modify it as follows: add a new initial neuron \(\sigma_{21}^{'}\) and \(\sigma_{21}\) is replaced with \(d\) neurons (instead of \(\sigma_{22}\)). The synapse set of \(\Pi\) is changed to \(\{(21', 21-1), \ldots, (21', 21-(d-1)), (21-1, 21-d), (21-d, 22), (22, 21-1), \ldots, (22, 21-(d-1))\}\), we remove \(\sigma_{23}\) and have \(\sigma_{22}\) as the output neuron instead. Both \(\Pi_3\) and \(\Pi_3\) halt at the same time at \(t + d + 2\).

![Figure 5: Iteration routing: \(\Pi_3\) (top) has a delay \(d\) simulated by \(\Pi_3\) (bottom).](image)

**Lemma 3** (Join routing). Given an SNP system with delay \(\Pi\) performing join routing, there exists an SNP system without delay \(\overline{\Pi}\) performing join routing that simulates \(\Pi\).

**Proof.** We refer to Figure 6 for illustrations. Let
\[
\Pi = (O, \sigma_{11}, \sigma_{12}, \sigma_{13}, \{(11, 13), (12, 13)\}, \Pi_3)\]
where \(\sigma_{11} = \sigma_{12} = (1, a^+/a \rightarrow a), \sigma_{13} = (0, (a^2)^+/a^2 \rightarrow a; d)\) we then let
\[
\overline{\Pi} = (O, \sigma_{21}, \sigma_{22}, \sigma_{23-1}, \sigma_{24}, \text{syn}, \Pi_4)\]
where \(\sigma_{21} = (1, a^+/a \rightarrow a), \sigma_{22} = (1, a^+/a \rightarrow a), \sigma_{23-1} = \ldots = \sigma_{23-(d-1)} = (0, (a^2)^+/a^2 \rightarrow a^2), \sigma_{23-d} = (0, (a^2)^+/a^2 \rightarrow a), \text{syn} = \{(21, 23-1), \ldots, (21, 23-(d-1)), (22, 23-1), \ldots, (22, 23-(d-1)), (23-1, 23-d), \ldots, (23-(d-1), 23-d), (23, 24-d)\}.

For \(\Pi\) and \(\overline{\Pi}\) we have as initial neurons \(\sigma_{11}, \sigma_{12}\) and \(\sigma_{21}, \sigma_{22}\) respectively. Using the construction in Lemma 1, \(\overline{\Pi}\) has \(d\) additional neurons corresponding to \(\sigma_{13}\) in \(\Pi\). Let time \(t\) be the time when the initial neurons spike. At time \(t\), neurons \(\sigma_{23-1} \text{ to } \sigma_{23-(d-1)}\) have two spikes each, so that in total, \(\overline{\Pi}\) at this time has \(2(d-1)\) spikes. At the next step \(t + 1\), the \(d-1\) neurons send two spikes each to \(\sigma_{23-d}\) so that \(\sigma_{23-d}\) accumulates \(2(d-1)\) spikes. Since \(\sigma_{23-d}\) consumes two spikes every time and it has \(2(d-1)\) spikes, \(\sigma_{23-d}\) will take \(d-1\) time steps to consume all of its \(2(d-1)\) spikes. Every time \(\sigma_{23-d}\) spikes, it sends only one spike to \(\sigma_{24}\). At time \(t + d\), \(\sigma_{24}\) has accumulated \(d-1\) spikes from \(\sigma_{23-d}\) so that \(\sigma_{24}\) sends one spike to the environment and halts at \(t + d + 1\). This time step coincides with the halting time of \(\sigma_{13}\) in \(\Pi\), sending one spike to the environment. We therefore satisfy \(R_1\) and \(R_2\).

A sample computation of \(\Pi_4\) and \(\Pi_4\) is shown in Table 3.

**Table 3:** Sample computations of \(\Pi_4\) and \(\Pi_4\), \(d = 3\).

| Steps | \(\Pi_4\) | \(\Pi_4\) |
|-------|----------|----------|
| \(t_0\) | \((1/0, 1/0, 0/0, 0)\) | \((1, 1, 0, 0, 0, 0)\) |
| \(t_1\) | \((0/0, 0/0, 2/0, 0)\) | \((0, 0, 2, 0, 0, 0)\) |
| \(t_2\) | \((0/0, 0/0, 3/0, 0)\) | \((0, 0, 0, 0, 4, 0, 0)\) |
| \(t_3\) | \((0/0, 0/0, 2/0, 0)\) | \((0, 0, 0, 2, 0, 0)\) |
| \(t_4\) | \((0/0, 0/0, 2/0, 0)\) | \((0, 0, 0, 0, 2, 0)\) |
| \(t_5\) | \((0/0, 0/0, 0/0, 0)\) | \((0, 0, 0, 0, 0, 0)\) |

**Lemma 4** (Split routing). Given an SNP system with delay \(\Pi\) performing split routing, there exists an SNP system without delay \(\overline{\Pi}\) performing split routing that simulates \(\Pi\).

**Proof.** We refer back to the split routing in Figure 2. Notice that a split routing can be thought of as two separate paths, either from \(\sigma_3\) to \(\sigma_4\) or \(\sigma_5\). Let \(t\) be the time that \(\sigma_3\) spikes and modify the split routing in Figure 2 as follows and let it be
\[
\Pi = (O, \sigma_3, \sigma_4, \sigma_5, \text{syn}, \Pi_4)\]
where \(\sigma_3 = (1, a^+/a \rightarrow a), \sigma_4 = (0, a^+/a \rightarrow a), \sigma_5 = (0, a^+\rightarrow a; d), \text{syn} = \{(3, 4), (3, 5), (4, a), (5, a)\}.

We arbitrarily chose \(\sigma_4\) to have a delay instead of \(\sigma_5\) in this case. Next we let
\[
\overline{\Pi} = (O, \sigma_3', \sigma_4', \text{syn}, \Pi_4)\]
where \(\sigma_3' = (1, a^+/a \rightarrow a), \sigma_4' = \ldots = \sigma_4'_{d-1} = \sigma_4'' = (0, a^+/a \rightarrow a), \sigma_4_{d-1} = (0, (a^{d-1})^+/a^{d-1} \rightarrow a), \text{syn} = \{(3', 4', 1), \ldots, (3', 4', (d-1)), (4', 1), 4', 4'-d), \ldots, (4', (d-1), 4'-d), (4'-d, a), (5, a)\}.

Since \(\sigma_d\) has delay \(d\), we simply follow Lemma 1 and add \(d\) neurons in \(\overline{\Pi}\) corresponding to \(\sigma_d\). Let \(t\) be the time when \(\sigma_3\) and \(\sigma_3'\) spike. The time that \(\sigma_3\) spikes the second time (since the spike from \(\sigma_3\) makes \(\sigma_3\) spike the first time, followed by the delayed spike from \(\sigma_3'\)) i.e. the halting time, is \(t + d + 1\) and the environment receives two spikes in total. This time coincides with the halting time of \(\overline{\Pi}\), which also sends two spikes to its environment. \(R_1\) and \(R_2\) are both satisfied for this case.
In the case where both \( \sigma_4 \) and \( \sigma_5 \) have delays \( d_4 \) and \( d_5 \) respectively, then following Lemma 1 \( \Pi \) has \( d_4 + d_5 \) additional neurons. For both systems, halting time is \( t + d_{max} + 1 \) where \( d_{max} = \max(d_4, d_5) \), and two spikes are sent to the environment, satisfying \( R_1 \) and \( R_2 \).

We can now have the following theorem.

**Theorem 1.** Given an SNP system \( \Pi \) with delays containing one or more of the following routings: sequential, iteration, join, split, there exists an SNP system \( \Pi' \) that simulates \( \Pi \).

**Proof.** Proof follows from Lemma 1, 2, 3, 4.

Notice that Observation 1 and 2 hold for all four routing constructs.

4. **FINAL REMARKS**

We have presented an alternative construction of a \( \Pi \) that simulates a given \( \Pi \), improving the previous work so that we use only one initial spike for both systems. The halting time of both systems also exactly coincide with one another. The trade off is that there is an “explosion” of neurons in \( \Pi \) for every delay in \( \Pi \) i.e. we add \( d_i \) neurons in \( \Pi \) for every \( \sigma_i \) that has a delay.

For our further work, we will consider nondeterministic SNP systems i.e. neurons having more than one applicable rule, since we only consider deterministic systems in this work. Minimization of the number of neurons of a \( \Pi \) simulating a \( \Pi \) is also desirable, including providing bounds to the number of neurons, spikes, and types of spiking rules. We will also use the matrix representation of SNP systems without delays from \[15\] and then use massively parallel processors such as graphics processing units to create simulations of computations as was done in \[2\]. Lastly, certain results and applications of SNP systems that use delays can be converted to SNP systems without delays e.g. generating automatic sequences as in \[3\], and performing arithmetic operations as in \[10\], among others.

5. **REFERENCES**

[1] Alhazov, A., Freund, R., Oswald, M., Slavkovic, M.: Extended Spiking Neural P Systems. Păun, G., Rozenberg, G., Salomaa, A. (eds) Membrane computing, international workshop, WMC7, revised, selected, and invited papers, Leiden, The Netherlands, LNCS, vol 4361. Springer, Berlin, pp 123-134 (2006)

[2] Cabarle, F.G.C., Adorna, H.N., Martínez-del-Amor, M.A., Pérez-Jiménez, M.J.: Improving GPU Simulations of Spiking Neural P Systems. Romanian Journal of Information Science and Technology, vol. 15(1) (2012)

[3] Cabarle, F.G.C., Buño, K.C., Adorna, H.N.: Time After Time: Notes on Delays In Spiking Neural P Systems. Workshop on Computation: Theory and Practice 2012 (WCTP 2012), 27-28 September, Manila, Philippines, preprint available online: http://arxiv.org/abs/1210.6119 (2012)

[4] Cabarle, F.G.C., Buño, K.C., Adorna, H.N.: Spiking Neural P Systems Generating the Thue-Morse Sequence. Asian Conference on Membrane Computing, Wuhan, China, Oct (2012)

[5] Chen, H., Ionescu, M., Ishdorj, T.-O., Păun, A., Păun, Gh., Pérez-Jiménez, M.J.: Spiking neural P systems with extended rules: universality and languages. Natural Computing, vol. 7, issue 2, pp. 147-166 (2008)

[6] Ibarra, O., Pérez-Jiménez, M.J., Yokomori, T.: On spiking neural P systems. Natural Computing, vol. 9, pp. 475-491 (2010)

[7] Ionescu, M., Păun, Gh., Yokomori, T.: Spiking Neural P Systems. Fundamenta Informaticae, vol. 71, issue 2,3 279-308, Feb. (2006)

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

[9] Pan, L., Păun, Gh., Pérez-Jiménez, M.J.: Spiking neural P systems with neuron division and budding. Proc. of the 7th Brainstorming Week on Membrane Computing, RGNC, Sevilla, Spain, pp. 151-168 (2009)

[10] Pan, L., Zeng, X., Zhang, X., Jiang, Y.: Spiking Neural P Systems with Weighted Synapses. Neural Processing Letters 35(1): 13-27 (2012)

[11] Păun, G.: Membrane Computing: An Introduction. Springer (2002)

[12] Păun, G., Pérez-Jiménez, M.J.: Spiking Neural P Systems. Recent Results, Research Topics. A. Condou et al. (eds.), Algorithmic Bioprocesses, Springer (2009)

[13] Păun, G., Rozenberg, G., Salomaa, A. The Oxford Handbook of Membrane Computing, Oxford University Press (2010)

[14] Zeng, X., Zhang, X., Pan, L.: Homogeneous Spiking Neural P Systems. Fundamenta Informaticae vol 97(1-2), pp. 275-294 (2009)

[15] Zeng, X., Adorna, H.N., Martínez-del-Amor, M.A., Pan, L., Pérez-Jiménez, M.J.: Matrix Representation of Spiking Neural P Systems. 11th CMC, Jena, Germany, Aug. 2010 and LNCS 6501, pp. 377-39 (2011)

[16] Zeng, X., Song, T., Zhang, X., Pan, L.: Performing Four Arithmetic Operations with Spiking Neural P Systems. IEEE Transaction on NanoBioscience. (in press)