Numerical P Systems: Variants and Applications

Radu Traian Bobe[1,*,†], Marian Gheorghe[2,†], Florentin Ipate[1,†] and Ionuţ Mihai Niculescu[1,†]

[1] *Department of Computer Science, Faculty of Mathematics and Computer Science University of Bucharest, Str Academiei 14, Bucharest, 010014, Romania*

[2] *Faculty of Engineering and Informatics, University of Bradford, Bradford, West Yorkshire, BD7 1DP, United Kingdom*

Abstract

Numerical P systems are compuational models which are inspired by cellular processes and use numerical values inside the membrane structure. In this paper we will present the variants of numerical P systems that have been proposed in the literature. In particular, we will discuss the advantages introduced by these extensions in terms of Turing completeness and potential applications. Moreover, using a reference example, we will experiment a mapping between some important numerical P systems variants.

Keywords

Membrane Computing, Numerical P systems, Spiking Neural P systems, Enzymatic Numerical P systems, Modelling

1. Introduction

The innovative and precise character of natural processes, often referred to as "the intelligence of matter" has led to the emergence of an interdisciplinary area at the intersection of biology and computer science, called natural computing. Membrane computing, evolutionary computation, cellular automata, as well as neural computing are just a few of the most important areas of natural computing.

Membrane computing is a branch of natural computing, developed by Gh. Păun in 1998 [1]. The concept is inspired by the biological functionality of living cells, abstracting computational models. A cell-like membrane system, also called a P system is a formalism that abstracts and mimics the processes observed in living cells, specifically focusing on how membranes within cells interact and process information. The membranes are represented as compartments that encapsulates objects and rules. We can refer to objects as entities that inhabit inside membranes and are transformed according to the rules. A rule is similar to a chemical reaction, to the extent that dictates the evolution of objects within and across membranes. Continuing the analogy with the bio-chemical domain, this process is similar to molecular interactions and transport across the cells.

Different variants of the original model were then proposed, based on the interactions and positioning of the cells. If *cell-like P systems* feature a hierarchical configuration of membranes as in a cell, *tissue-like P systems* [2] have several one-membrane cells arranged as nodes in an undirected graph. This arrangement gave the model name, because the cells are organized as in a tissue. Moreover, Ionescu et al. introduced in 2006 *neural-like P systems*, that incorporates the idea of spiking neurons into the area of membrane computing. [3]. This novel category of membrane systems has cells represented as neurons from a neural net. The communication between them is represented as electrical impulses called spikes. More definitions about membrane computing, as well as examples and technical results can be found in [4].

A category of cell-like P systems that use numerical values in the compartments sparked interest and innovations in fields like economics or robotics. These membrane systems are called *numerical P sytems* [5] and focus on manipulating numerical values inside the membrane structure. This paper aims to present the variants of numerical P systems. In particular, we will highlight the main differences between them, analysing the computational power. We will also demonstrate a transformation between the most interesting models proposed in the literature, taking the examples from the reference articles. This paper is structured as follows: Section 2 introduces the formal definition of numerical P systems and spiking numerical P systems. Section 3 presents and classifies all the numerical P systems variants. Section 4 illustrates the conversion from a numerical P system to a numerical P system with Boolean condition and a numerical spiking neural P system. Section 5 states the conclusions of this paper as well as future research directions.

ITAT'24: Information technologies – Applications and Theory, September 20–24, 2024, Drienica, Slovakia

† These authors contributed equally.

radu.bobe@s.unibuc.ro (R. T. Bobe); m.gheorghe@bradford.ac.uk (M. Gheorghe); florentin.ipate@unibuc.ro (F. Ipate); ionutmihainiculescu@gmail.com (I. M. Niculescu)
https://www.bradford.ac.uk/staff/mgheorghe (M. Gheorghe); https://www.ifsoft.ro/~florentin.ipate/ (F. Ipate); https:\ionutmihainiculescu.ro (I. M. Niculescu)
0009-0005-6611-3176 (R. T. Bobe); 0000-0002-2409-4959 (M. Gheorghe); 0000-0001-8777-3425 (F. Ipate); 0000-0002-6135-9135 (I. M. Niculescu)

2. Preliminaries

2.1. Numerical P System Definition

Before presenting any variant of numarical P system, we find it useful to introduce the formal definition of numerical P systems, the central concept of our survey. Without reiterating any details about membrane computing, we present a numerical P system [6] as the following tuple:

$$\Pi = (m, H, \mu, (Var_1, Pr_1, Var_1(0)), \dots, (Var_m, Pr_m, Var_m(0))) \tag{1}$$

where:

- $m \geq 1$ is degree of the system Π (the number of membranes);
- H is an alphabet of labels;
- μ is membrane structure;
- Var_i is a set of variables from membrane $i, 1 \leq i \leq m$;
- $Var_i(0)$ is the initial values of the variables from membrane $i, 1 \leq i \leq m$;
- Pr_i is the set of programs from membrane $i, 1 \leq i \leq m$.

  The membrane structure μ is a hierarchical arrangement of membranes that can be visualized as an expression of matching brackets, each pair representing a membrane. It also organizes inter-membrane communication [5] . The programs are the components responsible for computing the values of the variables at each simulation step. A program $Pr_{l_i,i}$, $1 \leq l_i \leq m_i$ has the following form:

  $$F_{l_i,i}(x_{1,i}, \dots, x_{k,i}) \rightarrow c_{1,i}|v_1 + c_{2,i}|v_2 + \dots + c_{m_i,i}|v_{m_i}$$

  where $F_{l_i,i}(x_{1,i}, \dots, x_{k,i})$ is the production function, $c_{1,i}|v_1 + c_{2,i}|v_2 + \dots + c_{m_i,i}|v_{m_i}$ is the repartition protocol, and $x_{1,i}, \dots, x_{k,i}$ are variables from Var_i. Variables $v_1, v_2 \dots v_{m_i}$ can be from the membrane where the programs are located, and from its outer and inner compartments, for a particular membrane i. If a compartment contains more than one program, a common strategy is to execute all the programs in parallel. This is called *all-parallel mode.* More details about how the variables values are computed according to the programs are presented in [5].

A numerical P system evolves by iteratively applying the rules that transform numerical values according to the dynamics defined. In this way, numerical processes observed in economics, robotics or real-word phenomena are simulated.
An example of numerical P system is presented later in this paper, in Section 4.1.

2.2. From Artificial Neural Networks to Spiking Neural P Systems

Artificial Neural Networks [7] are computational models inspired by the interaction between biological neural networks from the human brain. The central computing elements of an Artifical Neural Network (ANN) is called *neuron.* The connections between neurons, called *synapses* are made through input signals. The strength of these signals is controlled by the weights associated with synapses.
Spiking Neural Networks (SNNs) [8] are ANNs models of computation that mimic the functioning of neural networks. Unlike traditional ANNs, which use continuous activation functions, SNNs use discrete events known as "spikes" to transmit information. This leads to a more efficiently processing of time-dependent data.
Spiking Neural P systems (SN P systems) are a variant of membrane computing model that combine concepts from Spiking Neural Networks and membrane systems. Neurons are individual units that send out signals in form of spikes to other neurons. These spikes occur when the neuron accumulates enough input signals to reach a threshold. SN P systems were formalized by Ionescu et al. in [3].

During each computation step, the rules within each neuron are executed in parallel. If a neuron contains multiple rules, one of them will be nondeterministically chosen and applied. At any step, the configuration of the system is described by the states of the neurons represented by the quantity of spikes present in each neuron at that time.

The computing power of SN P systems is significant and represents a key component in the research area of this topic. It has been shown [3] that SN P systems are Turing complete which means that they can compute any computation that can be done if provided with enough memory and time. Moreover, SN P systems are powerful tools in biological modeling, as one of the main motivations for developing them was to understand and model biological neural networks.
An in-depth survey detailing the computational process of Spiking Neural P systems as well as the computational power of its variants can be consulted in [9].

3. Numerical P systems variants

The use of numerical values in the compartments of a cell-like membrane structure has naturally led to possible use cases in economics. Even if the economic interpretations of the variables evolution inside a numerical P system may vary, complex interactions between different currency exchange models or economic policies can be done.

Programs are fundamental to the functionality of NP systems and define computational processes, enabling powerful computing performance and numerical analysis. Starting from the importance of programs within a numerical P system and driven by the desire of better control the processes, numerous variants of NP systems have been proposed. In short, researchers have employed different mtehods of using the evolution programs to find new possible real-life applications of NP systems or to study the universality results as well as the computational power of the resulted variants.

In this section, we will briefly present the extensions introduced by each variant to the basic model, as well as some initial results and possible use cases.

In the classical model of a NP system, only one production function can be applied from each membrane in a time unit. This case is called *deterministic*. In the *stochastic* case, if a membrane contains more than one production function, one of them is randomly chosen. Enzymatic Numerical P systems (ENP systems) allow a better control of programs applications, by incorporating enzymes into the production functions, which allows for more accurate transformations of numerical values. As presented in [10], in an ENP system a rule can only be applied (in this case, the rule is called *active*) if the corresponding enzyme is present in the necessary amount. It is important to mention that an enzyme can be present in more than one production functions. All the active rules are then executed in parallel. In this way, more precise and regulated transformations of numerical values are obtained, reflecting the catalytic and regulatory roles of enzymes in biological systems. In addition to the use of numerical P systems in economics, enzymatic numerical P systems can be utilized in robotics to enhance the precision and adaptability of control algorithms [11].

Enzymatic numerical P systems were proved to be Turing universal, aspects like the complexity of polynomial production functions or the number of variables being investigated. The computational power on ENP systems was discussed in [12]. The topic was considered from a different point of view in [13], with the focus on determining the smallest number of enzymatic variables needed for universal ENP systems. Zhang et al. proved that if ENP systems are used as number acceptors in the all-parallel or one-parallel mode, only one enzymatic variable is needed to achieve universality. In the case of one-parallel ENP systems as number generators, two enzymatic variables are sufficient. The results significantly improved upon the previous data, where the numbers of enzymatic variables were 13 and 52 for the all-parallel and one-parallel systems, respectively.

Numerical P systems with thresholds (TNP systems) [14] employ a similar strategy to the *enzymatic control* in the sense that they use evolution programs in controlled manner. The difference is that a program can be applied only when the values of the variables involved in the production function are not smaller (Lower Threshold Numerical P systems) or not greater (Upper Threshold Numerical P systems) than the constant. A related approach was introduced in [15] where a *production threshold control* strategy is implemented. Instead of comparing every value of variables involved in the program with a constant, in numerical P systems with production thresholds (PTNP systems), the entire production value is compared with a constant. The production function is applied only when its value is not smaller (the lower-threshold case) or not greater (the upper-threshold case) than its associated constant. Even if the usability of these two numerical P systems in concrete applications remains an open subject, the language generating power of numerical P systems with thresholds was discussed in [16].

Numerical P systems with thresholds were the fundamental concept for other research topics as well, with their integration with Petri Nets being investigated in [26], where the operations of Petri Nets were associated with the evolutions in TNP systems. Another interesting approach is presented in [17] where six mall universal function computing devices of TNP systems were constructed.

Taking into consideration the above mentioned findings, it is unanimously accepted that control conditions play a crucial role in controlling the evolution of numerical P systems. Addressing potential practical applications, the large majority of dynamic systems requires an accurate control. The above mentioned numerical P systems variants have introduced the concept of control conditions. However, this conditions tend to have a simple logic and may not be able to achieve the requirements in a real-world dynamic system scenario. In order to overcome this limitation, Liu et al. [18] introduced the control condition in Boolean form and proposed a new variant of NP systems, called Numerical P Systems with Boolean condition (BNP systems). Even if the control condition in the previous variants of numerical P systems can be expressed as a Boolean condition, there were still a lot of Boolean expressions that cannot be expressed in all these variants. In a BNP system, the condition can be any Boolean expression using relational operators and linked by logical operators. In this way, BNP systems have introduced a stronger control mechanism that can be useful in real-world applications of NP systems. Moreover, it was proved that BNP systems are Turing universal as number generating/accepting devices and function computing devices, respectively, working in all-parallel, one-parallel and sequential mode.

In addition to these variants that target control conditions, other extensions of numerical P systems have also been proposed. Pavel and Dumitrache [19] introduced Hybrid Numerical P systems (HNP systems). Numerical P systems with migrating variables (MNP systems) [14] were inspired by the fact that in standard P systems an object can pass through membranes, between regions of the same cell, between cells, or between cells and their environment. In 2020, Yang et al. proposed another extension of NP systems, called Stochastic numerical P systems (StNP systems) [20]. The main difference arises from the stochastic production function-reparticion protocol, obtaining a class of distributed parallel computing models with applications in data clustering problems. Another interesting approach was proposed in [21], where a MIMD (Multiple Instruction Multiple Data) architecture is used to parallelise the elements of a NP system. Also, the generative capacity of numerical P systems as language generators was investigated in [22].

In the following, we will focus on P systems extensions obtained by combining the advantages offered by numerical P systems and spiking neural P systems (SNP systems). One of the main advantages of numerical P systems remains the use of numerical values as data representation. At the same time, SN P systems combine the strengths of membrane computing and spiking neural networks, the temporal aspect being one of the most attractive features. In 2020, Wu et al. [23] introduced numerical spiking neural P systems (NSN P systems), a new class of membrane systems that combines the advantages of NP systems and SNP systems. Similar to SNP systems, NSN P systems have a network architecture with an enhanced capability of representing complex topologies. Also, the information is encoded using numerical variables, involved in production functions that determine how the variables within the neurons evolve over time. The repartition protocol involves that after a production function in a neuron is executed, the resulted value is sent to all the variables present in adjacent neurons. Each production function can have a threshold and will be executed only when each value of the variables involved in it is not smaller than the threshold. The formal definition of NSN P systems is the following:

$$\mathit{\Pi} = (\sigma_{\mathit{1}}, \ldots, \sigma_m, syn, in, out) \tag{2}$$

where:

- $\sigma_{\mathit{1}}, \ldots, \sigma_m$ represent the *neurons* of the form $\sigma_i = (\mathit{Var}_i, \mathit{Prf}_i, \mathit{Var}_i(\mathit{0}))$, $\mathit{1} \leq i \leq m$, where:
  - (a) $\mathit{Var}_i = \{x_{q,i} \mid 1 \leq q \leq k_i\}$ represents the set of variables present in σ_i, where $x_{\mathit{1},i} \ldots x_{k,i}$ can be viewed as components of a n-dimensional real space vector
  - (b) $\mathit{Var}_i(0) = \{x_{q,i}(0) \mid x_{q,i}(0) \in \mathbb{R}, 1 \leq q \leq k_i\}$ refers to the set of initial values of the corresponding variables from Var_i
  - (c) Prf_i represents the set of production functions associated with each neuron
- *syn* is the set of synapses for each $(i, j) \in syn$, $\mathit{1} \leq i, j \leq m, i \neq j$
- *in, out* are the input and output neurons

Turing universality of NSN P systems has also been demonstrated in [23]. In terms of potential applications, NSN P systems can be suitable for applications that involve numerical information. Moreover, adding the threshold have made NSN P systems a powerful tool for applications that require deterministic mechanisms. In 2022, Jiang et al. [24] considered working with NSN P systems in asynchronous manner. They also proposed an extension of the traditional threshold strategy, introducing an extended threshold strategy which involves using a *threshold interval.* More precisely, a production function can be executed only if all of the variables involved are within the range of the threshold interval. In this way, the obtained asynchronous numerical spiking neural P systems (ANSN P systems) will be more flexible. The Turing universality as well as the computing power of ANSN P systems has also been discussed in [24].
The extension of NSN P systems is an active research direction, with concrete results in recent years. In this context, Yin et al. [25] introduced in 2021 a new variant of NSN P systems, called Novel numerical spiking neural P systems with a variable consumption strategy (NSNVC P systems). This new variant has led to improvements in the production functions, values of the variables involved having prescribed consumption rate without all being set to 0 after the execution. NSNVC P systems also use polarization of the neurons in order to control the execution of a production function.
Another variant that aims to improve the flexibility of the system has been introduced by Xu et al. [26] in 2023 by adding weights into NSN P systems. The resulted numerical spiking neural P systems with weights (NSNW P systems) are still Turing universal and the their computational power was demonstrated using fewer neurons than NSN P systems.
An interesting approach was proposed in [27]. While

in the classic NSN P systems the production functions are placed inside the neurons, in numerical spiking neural P systems with production functions on synapses (NSNFS P systems), the production functions of each neuron are placed on synapses. In this way, a neuron will contain only numerical variables. Potential applications of NSNFS P systems are further investigated.
A more practical initiative was presented by Zhang et. al in [28], where enzymatic numerical spiking neural membrane systems (ENSNP systems) were introduced. In the aforementioned paper, the practicality of EN systems is demonstrated by modelling ENSNP memb controllers for robots implementing wall following.

4. Illustrative Examples

4.1. Numerical P System

In this section we will illustrate two numerical P syst variants with similar evolution, using as starting point an example of numerical P system. By analysing the evolution of the proposed NP system, we designed two entities of different NP systems variants, achieving similar functioning. We chose to use a well-known NP system example, extracted from the article that introduced the concept of numerical P systems [5]. Let us introduce the system Π_1, having three membranes, each of them containing one variable. Examining the structure of Π_1 presented in Figure 1, we can observe that the rules suggest a sequential dependency where the variable values of each membrane influences the next. All membranes start with an initial value of zero for the variables defined inside them.
Analysing the system rules and starting with the third membrane, we can note that variable $x_{1,3}$ increases by 1 at each simulation step. The value of $x_{1,3}$ is also transmitted to $x_{1,2}$, as assigned in the repartition protocol.
In membrane 2, the value $2x_{1,2} + 1$ is transmitted to $x_{1,1}$. Also, it can be observed that the value of $x_{1,1}$ is never consumed, hence its value increases continuously. Taking this observation into account, we can see that the value of the targeted variable $x_{1,1}$ is constructed at each simulation step using the following algebraic identity: $n^2 = (n-1)^2 + 2(n-1) + 1$, $n > 0$. Table 1 contains the values of the variables after n = 4 simulation steps.
As stated before, we chose this example of numerical P system and next we will present two mappings into different numerical P systems variants, emphasizing the characteristics of each one.

Table 1
Evolution of NP system Π_1

Simulation step n	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
0	0	0	0
1	1	1	1
2	4	2	2
3	9	3	3
4	16	4	4

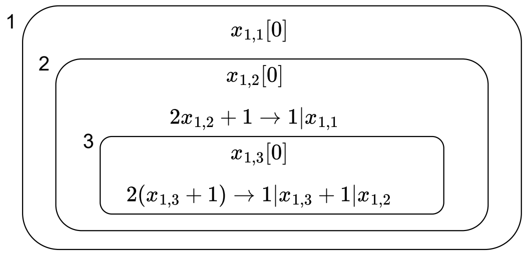
1
$x_{1,1}[0]$
2
$x_{1,2}[0]$
$2x_{1,2} + 1 \rightarrow 1|x_{1,1}$
3
$x_{1,3}[0]$
$2(x_{1,3} + 1) \rightarrow 1|x_{1,3} + 1|x_{1,2}$

Figure 1: Illustration of the numerical P system Π_1

4.2. Numerical P system with Boolean condition

As presented in Section 3, by integrating Boolean conditions into NP systems, the control over the computation process is improved, making selective rule application one of the main advantages of using Numerical P systems with Boolean condition (BNP systems). Thus, we will proceed with the modelling of a BNP system that calculates the perfect squares starting with 1, similar to Π_1. Moreover, by introducing Boolean conditions, we will store the even and the odd values of perfect squares in different membranes, using the same number of membranes and variables as the initial NP system presented above.
Looking at the resulted BNP system Π_2 which is illustrated in Figure 2, we can observe that the number of membranes is the same as in Π_1. Compartment 3 contains three programs. The first program is increasing the value of the variable $x_{1,3}$. The next two programs distribute the value needed to obtain the next odd/even perfect square that is calculated inside $x_{1,1}$, respectively $x_{1,2}$. Both the programs have Boolean conditions based on the value of $x_{1,3}$ and at each simulation step only one of them will be executed. It is important to mention that the obtained BNP system works in the *all-parallel* mode, executing all the applicable rules at each step. The distribution of the accumulated values into separate variables for odd and even perfect squares is based on a formula that is similar to the one used for Π_1: $n^2 = (n-2)^2 + 4(n-1)$, $n > 0$. The formula

Table 2
Evolution of BNP system Π_2

Simulation step n	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$
0	1	0	0
1	1	0	1
2	1	4	2
3	9	4	3
4	9	16	4

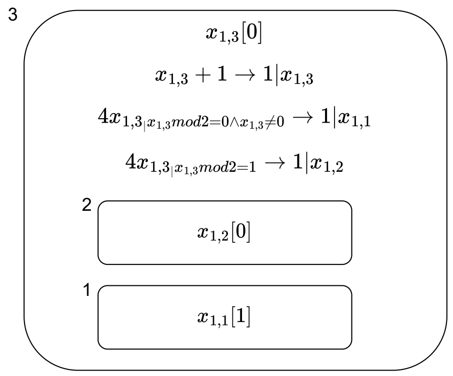
3
$x_{1,3}[0]$
$x_{1,3} + 1 \to 1|x_{1,3}$
$4x_{1,3|x_{1,3} mod 2=0 \wedge x_{1,3}\neq 0} \to 1|x_{1,1}$
$4x_{1,3|x_{1,3} mod 2=1} \to 1|x_{1,2}$
2
$x_{1,2}[0]$
1
$x_{1,1}[1]$

Figure 2: Illustration of the BNP system Π_2

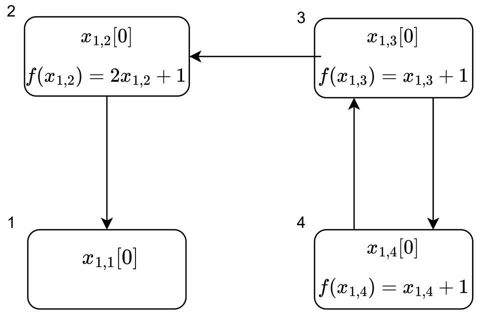
2
$x_{1,2}[0]$
$f(x_{1,2}) = 2x_{1,2} + 1$
3
$x_{1,3}[0]$
$f(x_{1,3}) = x_{1,3} + 1$
1
$x_{1,1}[0]$
4
$x_{1,4}[0]$
$f(x_{1,4}) = x_{1,4} + 1$

Figure 3: Illustration of the NSN P system Π_3

Table 3
Evolution of NSN P system Π_3

Simulation step n	$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
0	0	0	0	0
1	1	1	1	1
2	4	2	2	2
3	9	3	3	3
4	16	4	4	4

can be observed by analysing how $x_{1,1}$ and $x_{1,2}$ accumulate values at each simulation step, based on the values obtained at the previous steps. The first simulation steps are presented in Table 2. We note that the values of variables $x_{1,1}$ and $x_{1,2}$ should be considered just for odd, respectively even values of n, as they change at different steps.

4.3. Numerical Spiking Neural P System

In order to obtain another membrane system with the same functionality as Π_1, we designed Π_3, a NSN P system looking for the set of values taken by variable $x_{1,1}$, which is never consumed. Moreover, we can observe that Π_3 follows the same recursive formula as Π_1, $x_{1,1}$ receiving $2(n-1)+1$ at each simulation step *n*. The result is graphically presented in Figure 3.

The NSN P system illustrated in Figure 3 consists of four neurons, each containing one variable with an initial value of zero. Neurons $\sigma_2, \sigma_3, \sigma_4$ also contain the production functions that will be executed during the evolution of the system. It is important to mention that the first index of each variable represents the order of the variable within the neuron (as we have only one variable in each neuron, the first index will be 1 for all the variables), whilst the second index represents the label of the neuron. However, the target variable remains $x_{1,1}$, whilst $x_{1,2}$ and $x_{1,3}$ maintain similar roles as in Π_1. Table 3 illustrates the first simulation steps of the NSN P system obtained.

Initially, the value of variable $x_{1,3}$ is zero and the production function from neuron σ_3 is executed. In this way, neuron σ_3 transmits a value of 1 to neurons σ_2 and σ_4. Looking at the production function from the neuron σ_3, it can be observed that it does not replicate the original production function from the membrane 3 of the NP system Π_1, even if neuron σ_3 seems to be the correspondent of membrane 3. This aspect can be explained by the repartition protocol of the program from membrane 3, dividing the value between $x_{1,3}$ and $x_{1,2}$. According to the distribution stage of a NSN P system, the computed production value is transmitted to each variable from all the neighboring neurons, so there is no need to divide it. After executing the production function, the value of $x_{1,3}$ is reset to zero.
Returning to the analysis of the system, we note that neuron σ_3 receives 1 from neuron σ_4. Neuron σ_4 is used as an auxiliary structure, implementing the behavior of the program inside membrane 3 of the original NP system, where the variable $x_{1,3}$ appears both in the production function and the repartition protocol. In order to achieve this functionality in the NSN P system,

we used neuron σ_4 to transmit the updated value of the variable after being calculated.
At the same time, neuron σ_2 transmits a value of 1 to variable $x_{1,1}$. As σ_1 does not contain any production function, the value of $x_{1,1}$ will be accumulated.

One can observe that Π_1 and Π_3 use the same recursive formula, $n^2 = (n-1)^2 + 2(n-1) + 1$, where $(n-1)^2$ is stored in $x_{1,1}$ of compartment 1 and $2(n-1)+1$ is computed in compartment 2 of each of them. The second model, Π_2, a BNP system, computing two sets of perfect squares, uses another recurrence, $n^2 = (n-2)^2 + 4(n-1)$, where $(n-2)^2$ is stored either in compartment 1 or 2, depending on whether n is odd or even, respectively, and $4(n-1)$ is computed in compartment 3, the only compartment containing programs. Π_3 has a slightly more complex architecture, with more compartments and implicitly programs and variables.

Even for a relatively simple example, one can notice slight variations in providing models with three different numerical P systems. It is expected that more complex case studies may require more diverse set of features for each of the models involved, hence the need to find optimal ones, with respect to certain criteria, such as descriptional complexity measures (number of compartments, connections, rules and programs complexity).

5. Concluding remarks

This paper presented the main theoretical results concerning numerical P systems and their variants. We briefly described each computation model, highlighting the innovations and potential applications. By exploring these extensions of the basic concept, the reader can consider using a targeted model adapted to specific requirements. We also used a numerical P system with a concise structure to illustrate the mapping to numerical P systems with Boolean condition (BNP systems) and numerical spiking neural P systems (NSN P systems). The results highlighted the main improvements introduced by each variant.
As further developments, we will include more computation models in our mappings, also considering other examples. As a testing methodology [29] for SNP systems already exists, the mapping from NP systems to NSN P systems can serve as the starting point of a promising research line to develop a testing methodology for NSN P systems. Furthermore, motivated by the applicability of numerical P systems in areas of significant importance, such as economics or robotics, we will elaborate a testing theory and some testing methods for the most relevant numerical P systems classes.

References

[1] Gh. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (2000) 108–143. doi:10.1006/jcss.1999.1693.

[2] C. M. Vide, J. Pazos, Gh. Păun, A. R. Patón, Tissue P systems, Theoretical Computer Science 296 (2003) 295–326. doi:10.1016/S0304-3975(02)00659-X.

[3] M. Ionescu, Gh. Păun, T. Yokomori, Spiking neural P systems, Fundamenta informaticae 71 (2006) 279–308.

[4] Gh. Păun, Membrane Computing: An Introduction, Springer Science & Business Media, 2002.

[5] Gh. Păun, R. Păun, Membrane computing and economics: Numerical P systems, Fundamenta Informaticae 73 (2006) 213–227.

[6] R. T. Bobe, F. Ipate, I. M. Niculescu, Modelling and search-based testing of robot controllers using enzymatic numerical P systems, arXiv preprint arXiv:2309.13795 (2023). doi:10.48550/arXiv.2309.13795.

[7] K. Gurney, An introduction to neural networks, CRC press, 2018. doi:10.1201/9781315273570.

[8] S. Ghosh-Dastidar, H. Adeli, Spiking neural networks, International journal of neural systems 19 (2009) 295–308. doi:10.1142/S0129065709002002.

[9] A. Leporati, G. Mauri, C. Zandron, Spiking neural P systems: main ideas and results, Natural Computing 21 (2022) 629–649. doi:10.1007/s11047-022-09917-y.

[10] A. Pavel, O. Arsene, C. Buiu, Enzymatic numerical P systems-a new class of membrane computing systems, in: International Conference on Bio-Inspired Computing: Theories and Applications, IEEE, 2010.

[11] A. B. Pavel, C. Buiu, Using enzymatic numerical P systems for modeling mobile robot controllers, Natural Computing 11 (2012) 387–393.

[12] C. I. Vasile, A. B. Pavel, I. Dumitrache, Gh. Păun, On the power of enzymatic numerical P systems, Acta Informatica 49 (2012) 395–412.

[13] Z. Zhang, T. Wu, A. Păun, L. Pan, Universal enzymatic numerical P systems with small number of enzymatic variables, Science China Information Sciences 61 (2018) 1–12. doi:10.1007/s11432-017-9103-5.

[14] Z. Zhang, L. Pan, Numerical P systems with thresholds, International Journal of Computers Communications & Control 11 (2016) 292–304.

[15] L. Pan, Z. Zhang, T. Wu, J. Xu, Numerical P systems with production thresholds, Theoretical Computer Science 673 (2017) 30–41.

[16] L. Zhang, F. Xu, Languages generated by numerical P systems with thresholds, Theoretical Computer

Science 988 (2024) 114376.

[17] L. Liu, W. Yi, Q. Yang, H. Peng, J. Wang, Small universal numerical P systems with thresholds for computing functions, Fundamenta Informaticae 176 (2020) 43–59.

[18] L. Liu, W. Yi, Q. Yang, H. Peng, J. Wang, Numerical P systems with boolean condition, Theoretical Computer Science 785 (2019) 140–149. doi:10.1016/j.tcs.2019.03.021.

[19] A. B. Pavel, I. Dumitrache, Hybrid numerical P systems 78 (2016) 139–146.

[20] J. Yang, H. Peng, X. Luo, J. Wang, Stochastic numerical P systems with application in data clustering problems, IEEE Access 8 (2020) 31507–31518.

[21] D. Reid, A. Oddie, P. Hazlewood, Parallel numerical P systems using a mimd based architecture, in: 2010 IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA), IEEE, 2010, pp. 1646–1653. doi:10.1109/BICTA.2010.5645257.

[22] Z. Zhang, T. Wu, L. Pan, On string languages generated by sequential numerical P systems, Fundamenta Informaticae 145 (2016) 485–509.

[23] T. Wu, L. Pan, Q. Yu, K. C. Tan, Numerical spiking neural P systems, IEEE Transactions on Neural Networks and Learning Systems 32 (2020) 2443–2457. doi:10.1109/TNNLS.2020.3005538.

[24] S. Jiang, Y. Liu, B. Xu, J. Sun, Y. Wang, Asynchronous numerical spiking neural P systems, Information Sciences 605 (2022) 1–14.

[25] X. Yin, X. Liu, M. Sun, Q. Ren, Novel numerical spiking neural P systems with a variable consumption strategy, Processes 9 (2021) 549.

[26] B. Xu, S. Jiang, Z. Shen, X. Zhu, T. Liang, Numerical spiking neural P systems with weights, Journal of Membrane Computing 5 (2023) 12–24. doi:10.1007/s41965-022-00116-3.

[27] S. Jiang, B. Xu, T. Liang, X. Zhu, T. Wu, Numerical spiking neural P systems with production functions on synapses, Theoretical Computer Science 940 (2023) 80–89.

[28] L. Zhang, F. Xu, D. Xiao, J. Dong, G. Zhang, F. Neri, Enzymatic numerical spiking neural membrane systems and their application in designing membrane controllers, International Journal of Neural Systems 32 (2022) 2250055.

[29] F. Ipate, M. Gheorghe, A model learning based testing approach for spiking neural P systems, Theor. Comput. Sci. 924 (2022) 1–16.