Chapter 1

Introduction

This thesis is devoted to the field of membrane systems which studies the potential of a computational model inspired by information processing in the cells and tissues, and in our case particular that of neurons. This first chapter introduces the basic notions involved and sketches the background of the topic, as was done in [14]. In its final section the remaining chapters is summarized.

1.1 Membrane Computing

Science develops fast, especially in the multi-disciplinary fields, and new research areas come out continuously. In the field of computer science, it is popular to employ new computing ideas from biology. Natural computing is a typical example, which includes well-known evolutionary computing, neural computing, DNA computing, and so on. All these now classic paradigms are inspired by biological systems [8].

Membrane computing, one of the youngest branches of natural computing, was first proposed by Gheorghe Păun in 1998, see [15]. Its aim is to abstract computing models from the structure and the functioning of cells in tissues or other higher order structures. The resulting models are called membrane systems, which are also called P systems to honour to the contribution of Păun. Since membrane systems have powerful capability in parallel processing and significant potential in various applications, more and more researchers begin to get interested in this area. Already in 2003, Thomson Institute for Scientific Information (ISI, for short) has qualified the initial paper as "fast breaking" and the domain as "emergent research front in computer science".

According to different organizations of cells which inspire the different computing models, P systems can be classified to three main types as follows: (i) Cell-like P systems inspired from living cells; (ii) Tissue-like P systems inspired from tissues; (iii) Neural-like P systems inspired from neural systems. In this thesis, we will study the subclass of neural-like P systems which is called spiking neural P systems (or SN P systems, for short). As all neural-like P systems, these systems have the network structure that is gleaned form the brain.

The human brain is a complex and enormous information processing system, where more than a trillion neurons work in a cooperative manner to perform various tasks. Therefore, the brain is a rich source of inspiration for informatics as natural computing proves. Inspired from the biological phenomenon that neurons cooperate in the brain by exchanging spikes via synapses, Păun et al. developed the SN P systems in 2006 (see [7]). In such models, all the spikes are viewed as indistinguishable and the abstract symbol "a" is used to represent a spike. Different from the traditional computing models, in SN P systems the information is encoded in a sequence of moments in time when the unique "symbol" occurs rather than a sequence of different "symbols". It has been proved that SN P systems are computationally complete and a characterization of semilinear sets can be obtained for the behaviour of SN P systems under certain restrictive conditions.

After the initial proposal of SN P systems, there have been over hundreds of papers published in this field, we refer to [17]. Here, we will briefly introduce the definition of standard SN P systems and shortly introduce several forms of SN P systems motivated by additional nature-inspired featurs, as well as related notions and main results on SN P systems.

1.2 Spiking Neural P Systems

Before starting to introduce the various "enhanced" forms of SN P systems as studied in this thesis, it is necessary to recall the SN P systems with standard rules. For simplicity, we will shortly refer to them as standard SN P Systems.

A spiking neural P system (an SN P system, for short), of degree $m \geq 1$, is a construct of the form

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

where the constituents are composed as follows.

1) The alphabet $O = \{a\}$ consists of a single element a, called *spike*, the basic unit of information;

2) Components $\sigma_1, \ldots, \sigma_m$ are *neurons*, specified in the form $\sigma_i = (n_i, R_i)$, where for $1 \leq i \leq m$,

   (1) $n_i \geq 0$ is the initial number of spikes contained in σ_i,

   (2) R_i is a finite set of rules that can be in one of two forms.

   ① Firing rules, written as $E/a^c \to a; d$ —where E is a regular expression over a specifying when the rule is enabled, parameter $c \geq 1$ the number of spikes consumed, and $d \geq 0$ is the delay time, that is, the interval between using the rule and releasing the spike. If $d = 0$, then d usulally is omitted from the firing rules.

The intended semantics of the firing rule is as follows. If the neuron σ_i contains k spikes, a^k is contained in the language $L(E)$ specified by expression E (this means that the regular expression E "covers" exactly the contents of the neuron) and $k \geq c$, then the firing rule $E/a \to a; d$ can be applied in this neuron, meaning that exactly c spikes are consumed (leaving $k-c$ spikes in neuron σ_i) and producing a spike after d time units. This new spike will be sent to all neurons which are connected to neuron σ_i. Neuron σ_i is "closed" during the interval between applying the rule and releasing the new spike, meaning it cannot receive any spike during that time.

② Forgetting rules take the form $a^s \to \lambda$, for some $s \geq 1$, with the restriction that $a^s \notin L(E)$ for any firing rule $E/a^c \to a; d$ in neuron σ_i (i.e., a firing rule and a forgetting rule cannot be enabled simultaneously in a neuron).

As suggested by its name the semantics of such rule determines that the involved spikes disappear form the system. If neuron σ_i contains exactly s spikes, then the forgetting rule $a^s \to \lambda$ can be applied (and by definition no firing rule can be enabled in this step). Applying the forgetting rule as above means s spikes are consumed, but no new spike is produced. These rules heve no delay.

3) The topology of the membrane system is given by relation $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$, indicating the *synapses* between neurons. No neuron is connected to itself, that is, for all $1 \leq i \leq m$, one has $(i, i) \notin syn$;

4) The value $i_0 \in \{1, 2, \ldots, m\}$ indicates the identity of the the *output neuron* σ_{i_0}, which can send spikes to the environment, indicating special events in the computation. The resulting "output" of the system is ususlly defined in terms of occurrences of the output neuron firing, as detailed below.

In what follows, we briefly recall some additional notions related to SN P systems.

In most of the literature on SN P systems, one considers SN P systems working in a synchronous manner. Under the synchronous mode, in each time unit, each neuron which can apply a rule should do it, but at most one rule can be applied in each neuron. This means that there is an implicit global clock involved in the system, and that firing of rules is globally maximally parallel (applying as many rules as possible in the system) but locally sequential (in each neuron).

Another notion which is often used in SN P systems is the spike train, and several ways of defining the final result of a computation are based on this notion. The moments of time when a spike is emitted by the output neuron are marked with 1, the moments of time when no spike is emitted are marked with 0. The produced binary sequence is called the spike train of the system. If a computation halts, meaning no firther rules are applicable, then its spike train is finite; if a computation does not halt, then its spike train is infinite.

Up to now, there have been several ways of defining the computational result of SN P systems:

In the initial papers on SN P systems, the result was often defined as the distance between the first two spikes of a spike train. Then in [3] several extensions were examined. The distance between the first k spikes of a spike train, or the distances between all consecutive spikes, taking into account all intervals or only intervals that alternate, all computations or only halting computations, etc.

The way one generally defines the result in membrane computing can also be considered in SN P systems, namely, the result is taken to be the total number of spikes collected in the output neuron (or the environment) during a computation at moment the computation halts.

Moreover, the result can obviously be defined as the spike train itself. In this way, SN P systems are used as binary language generators.

The SN P systems mentioned above all work in the generating mode: starting from a fixed initial configuration the nondeterministic features of the system make it run in different computations, and the output of these conputations is collected. Actually, an SN P system can also work in the accepting mode: no output neuron is needed, but intead an input neuron is designated. Then as an encoding of input n, two consecutive spikes are introduced in this input neuron, at an interval of n time steps; the number n is accepted if the resulting computation halts. These systems can be deterministic (meaning that for each input there is only one resulting computation) as opposed to the generating case, where determinism is not a very fruitful concept.

1.3 Several Other SN P Systems

After the standard SN P systems were proposed and investigated, several other SN P systems were developed by researchers. Generally, one distinguishes various families of SN P systems of which we present four. The first variant involves the type of rules. This is orthogonal to the next three variants which consider different forms of synchronization and parallellism in the system.

Extended SN P systems

In standard SN P systems, in each time step only one single spike can be produced at a given neuron by a firing rule. Several spikes can be produced simultaneously in extended SN P systems (see [3]). In many cases this feature makes the systems more flexible and more easy to “program”.

SN P systems with exhaustive use of rules

In the usual SN P system (either standard SN P system or extended as discussed above), although the system works in parallel at the level of all neurons (i.e., the system works in the synchronous manner), at the level of each neuron only (at most) one rule can be applied. In the SN P systems working in the exhaustive manner, when a rule is applied, then it must be applied as many times as possible in that neuron so that the neuron remains as few spikes as possible. Under the exhaustive mode, not only the system works in parallel at the level of all neurons, but also a rule is applied in a maximally parallel manner at the level of each separate neuron.

Asynchronous SN P systems

The SN P systems mentioned above each neuron which can apply one or several rules should apply such a rule. In SN P systems working in the asynchronous manner, in any step a neuron is not forced to apply any rule which are enabled by the number of spikes it contains. At the same time, further spikes can come into this neuron, thus changing the rules enabled in the next step. If the contents of the neuron are not changed, a rule which was enabled in a step t can fire later. If new spikes are received, then it is possible that other rules will be enabled – and applied or not (see [2]).

Sequential SN P systems

In sequential SN P systems, at every step of the computation, if there is at least one neuron with at least one rule that is firable, only one such neuron and one such rule is allowed to be fired, both chosen non-deterministically. Such systems, not only work sequentially at the level of all neurons, but also the rules are sequentially applied at the level of each neuron (see [5]).

1.4 Other Variants

Additional features inspired by nature involve axons, neuron division, and astrocytes. These are important concepts in the remainder of this thesis.

Axon P systems

In SN P systems the main "information-processor" is the neuron, while the axon is only a channel of communication without any other role – which is not exactly the case in neurobiology. So, recently, a special form of spiking neural P systems, called axon P systems, was introduced by Chen etal. in [4], which correspond to the activity of Ranvier nodes of neuron axon. Actually, axon P systems are a sort of linear SN P systems. The computational power of axon P systems was investigated in [4] and our [21]. As language generators, it was proved that the axon P systems are not computationally complete; as number generators, it can only be proved that axon P systems can generate a non-semilinear set.

SN P systems with neuron division and budding

The biological motivation for the mechanism of neuron division and budding that we introduce into SN P systems comes from the recent discoveries in neurobiology related to neural stem cells. Neural stem cells are persistent throughout life within central nervous system in the adult mammalian brain, which ensures a life-long contribution of new neurons to a self-renewing nervous system with about 30000 new neurons being produced every day. These observations are incorporated in SN P systems by considering neuron division and budding, and by providing a "synapse dictionary" according to which new synapses are generated, respectively. In [10], SN P systems with neuron division and budding solved computationally hard problems, where for all $n, m \in \mathbb{N}$ all the instances of $\mathtt{SAT}(n, m)$ with at most n variables and at most m clauses are solved in a deterministic way in polynomial time using a polynomial number of initial neurons.

SN P systems with anti-spikes

SN P system with anti-spikes introduced in [9], is a variant of an SN P system consisting of two types of objects, spikes and anti-spikes. The inhibitory impulses/spikes are represented using anti-spikes. The anti-spikes behave in a similar way as spikes by participating in spiking and forgetting rules. They are produced from usual spikes by means of usual spiking rules; in turn, rules consuming anti-spikes can produce spikes or anti-spikes (see [18]).

SN P systems with astrocytes

An important ingredient of neurobiology is missing from SN P systems: the astrocytes. They are cells which play an essential role in the functioning and interaction of neurons, by feeding them differently with nutrients depending on their individual activity. More specifically, astrocytes are cells which sense at the same time the spike traffic along several neighboring axons, and feed the respective neurons (e.g., with calcium) depending on the spikes frequency. The first attempt to introduce this kind of cells into SN P systems was made by Binder et al. in [1]. In this model, astrocytes have the role of "excitatory and inhibitory" according to the quantity of spikes passing through the synapse in one time. Also, a simple model is considered by Păun in [16], where astrocytes only have the role of "inhibitory". In such a model, the system works in a rather restricted manner: An astrocytes checking several axons leaves to pass only one spike along them, suppressing all others. Obviously. SN P systems with astrocytes have stronger computing power than the general SN P systems, thus they are also computationally complete. However they add a new powerful feature so that this kind of models has a significant potential to be applied to various problems in terms ease of programming and in terms of computational complexity.

1.5 A Simple Example

In order to show how an SN P system works, we give a simple example.

The system is shown in Figure 1.1. In this figure, each neuron is represented by an oval, marked with a label i indicating the identity of neuron σ_i, and having inside both the current number of spikes and the evolution rules for that neuron. The synapses linking the neurons are represented by directed edges; in addition, there is an edge from neuron σ_3 to the environment, which indicates that this neuron is the output one.

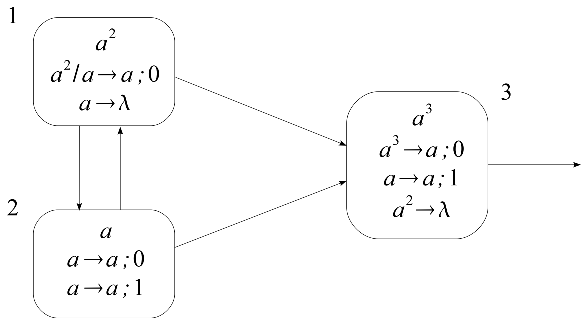

Figure 1.1: A simple example

In this example it is assumed that the system works in the synchronous manner and at most one rule is applied in one time at the level of each neuron.

From Figure 1.1 we see that, in the beginning, neuron σ_1 contains two spikes, σ_2 contains one spike and σ_3 contains three spikes. Therefore, in neuron σ_1, the rule $a^2/a \to a; 0$ can be applied; in neuron σ_2, one of the two rules $a \to a; 0$ and $a \to a; 1$ can be applied, non-deterministically chosen; in neuron σ_3, the rule $a^3 \to a; 0$ can be applied. Assume that the rule $a \to a; 0$ is applied in neuron σ_2, then one spike is sent to neurons σ_2 and σ_3, from neuron σ_1, one spike is sent to neurons σ_1 and σ_3 from neuron σ_2, and one spike is sent to the environment from neuron σ_3. Thus, the three neurons contain two, one and two spikes, respectively. If we continue to use the rule $a \to a; 0$ in neuron σ_2, then the computation will continue (neuron σ_3 uses its forgetting rule $a^2 \to \lambda$ during this process). Assume that at some step the rule $a \to a; 1$ is applied in neuron σ_2, then in the next step one spike is sent to both neuron σ_1 and neuron σ_3 from neuron σ_2. Note that this neuron cannot receive any spike in this step (it is closed). Hence, in this case, the three neurons contain one spike, no spike and one spike, respectively. In this way, the rule $a \to \lambda$ can be applied in neuron σ_1, and the rule $a \to a; 1$ can be applied in neuron σ_3, thus another spike is sent to the environment. In this step, neuron σ_2 will also send a spike to both neuron σ_1 and neuron σ_3. In neuron σ_1, the spike is forgotten by using the rule $a \to \lambda$; in neuron σ_3, because the rule $a \to a; 1$ is applied in this step, it cannot receive any spike.

Therefore, the system contains no spike and thus no rule can be applied in any of the neurons, which means that the computation halts.

In this example, we define the result of a computation as the distance between the first two spikes of a spike train. As explained above, at any moment, once the rule $a \to a; 1$ is applied in neuron σ_2, the system will send the second spike to the environment after delaying one step and then the computation halts. Hence, the system can generate $N_2(\Pi) = \mathbb{N}^+ - \{1\}$, where $\mathbb{N}^+$ is the set of all positive integers (number 0 is ignored, for technical reasons: it cannot be generated by these systems).

1.6 Overview of the Thesis

The chapters in this book are based on individual papers [6, 11, 12, 13, 19, 20, 21] co-authored by the author of this thesis. We give a short introduction to each of them.

In Chapter 2, we solve the open problem from [2] whether asynchronous SN P systems with standard rules are universal. In a biological system, given a long enough time interval, an enabled chemical reaction will finish its reaction within the time interval. So, it is natural to impose a bound on the time interval in which a spiking rule remains unused. In an SN P system working in limited asynchronous mode, if a rule is enabled at some step, this rule is not obligatorily used in the same step. However, if from this moment on the rule remains applicable, it should be used in the given time interval. If further spikes arriving in the neuron make the rule non-applicable, then the computation continues in the new circumstances. We prove that limited asynchronous SN P systems with standard spiking rules are universal. This chapter is based on [12].

In Chapter 3, we introduce SN P systems with neuron budding rules, a new feature that enhances the efficiency of SN P systems by allowing them to generate an exponential size synapse graph. We have shown that a very restricted type of neuron budding rules, involving one or two synapses is sufficient to solve the `SAT` problem: the first phase builds an exponential size SN P system that contains no spikes; then, this SN P system is fed with the instance of `SAT` to be solved and the answer is computed. This chapter is based on [6].

In Chapter 4, to answer the question posed in [10], a uniform family of SN P systems with only neuron division is constructed for efficiently solving the `SAT` problem. We improve the result of [10] in the sense that the SN P systems are constructed with a constant number of initial neurons instead of linear number with respect to the parameter n, while the computations still last a polynomial number of steps. This chapter is based on [19].

In Chapter 5, we use the axon P systems both as number generators and language generators. As a number generators, we consider as the result of a computation the number of steps elapsed between the two moments when the output node spikes. In this case, a characterization of the family of finite sets is given by axon P systems with one node, and the relationship of sets of numbers generated by axon P systems with semilinear sets of numbers is also investigated. As a language generator, the result of a halting computation is defined as the strings associated with configurations where the system emits a spike. In this case, the relationships of the families of languages generated by axon P systems with finite and context-free languages are considered. This chapter is based on [21].

In Chapter 6, we introduce a variant of SN P systems with weighted synapses, where the potentials of neurons are processed by spiking rules, and the applicability of spiking rules is under the control of given firing thresholds. The involved values – weights, firing thresholds, potential consumed by each rule – can be either real (computable) numbers, rational numbers, integers, or natural numbers. It is proved that integers suffice for computing all Turing computable sets of numbers in both the generative and the accepting modes. When natural numbers are used for weights, potentials, and thresholds, a characterization of semilinear sets of natural numbers is obtained. It is shown that spiking neural P systems with weights can efficiently solve computationally hard problems in a non-deterministic way. This chapter is based on [20].

In Chapter 7, we present a variant of SN P systems with astrocytes, where an astrocyte can sense at the same time the spike traffic along several neighbouring synapses. With the excitatory and inhibitory role of astrocytes, it is proved that SN P systems with astrocytes are universal even with simple neurons (each neuron has only one spiking rule) while also all neurons can be homogeneous. If a bound is given on the number of spikes present in any neuron along a computation, a characterization of semilinear sets of numbers is obtained. This chapter is based on [13].

In Chapter 8, we prove that spiking neural P systems with astrocytes are equivalent with Turing machines in the non-synchronized way: if a rule is enabled at some step, this rule is not obligatorily immediately used. Further spikes may make the rule non-applicable. This chapter is based on [11].

Bibliography

[1] A. Binder, R. Freund, M. Oswald, and L. Vock. Extended spiking neural P systems with excitatory and inhibitory astrocytes. *Proceedings of the 8th Conference on 8th WSEAS International Conference on Evolutionary Computing*, Vancouver, Canada, pages 320–325, 2007.

[2] M. Cavaliere, O. Egecioglu, O. H. Ibarra, M. Ionescu, Gh. Păun, and S. Woodworth. Asynchronous spiking neural P systems: decidability and undecidability. *Proceedings of the 13th International Meeting an DNA Computing (DNA 13)*, Memphis, Tennessee, 2007. *Lecture Notes in Computer Science* 4848, Springer-Verlag, 246–255, 2008.

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

[4] H. Chen, T.-O. Ishdorj, and Gh. Păun. Computing along the axon. *Progress in Natural Science*, 17(4): 417–423, 2007.

[5] O. H. Ibarra, S. Woodworth, F. Yu, and A. Păun. On spiking neural P systems and partially blind counter machines. *Neural Computing*, 7(1): 3–19, 2008.

[6] T.-O. Ishdorj, A. Leporati, L. Pan, and J. Wang. Solving NP-complete problems by spiking neural P systems with budding rules. In: Gh. Păun, M.J. Pérez-Jiménez and A. Riscos-Núñez (Eds.), *Proceedings of the Tenth Workshop on Membrane Computing*, pages 514–533, University of Seville, 2009.

[7] M. Ionescu, Gh. Păun, and T. Yokomori. Spiking neural P systems. *Fundamenta Informaticae*, 71(213): 279–308, 2006.

[8] L. Kari and G. Rozenberg. The many facets of natural computing. *Communications of the ACM*, 51(8): 72–83, 2008.

[9] L. Pan and Gh. Păun. Spiking neural P systems with anti-spikes. *International Journal of Computers, Communications and Control*, 4, 273–282, 2009.

[10] L. Pan, Gh. Păun, and M.J. Pérez-Jiménez. Spiking neural P systems with neuron division and budding. *Seventh Brainstorming Week on Membrane Computing*, vol. II, 151–168, 2009.

[11] L. Pan, J. Wang, and H.J. Hoogeboom. Asynchronous extended spiking neural P systems with astrocytes. In: M. Gheorghe, Gh. Păun, and S. Verlan (Eds.), *Twelfth International Conference on Membrane Computing*, pages 299–314, 2011. To appear in Lecture Notes in Computer Science.

[12] L. Pan, J. Wang, and H.J. Hoogeboom. Limited asynchronous spiking neural P systems. *Fundamenta Informaticae*, 110(1-4): 271–293,, 2011.

[13] L. Pan, J. Wang, and H.J. Hoogeboom. Spiking neural P systems with astrocytes. *Neural Computation.* Accepted.

[14] L. Pan, X. Zhang, X. Zeng, and J. Wang. Research advances and prospect of spiking neural P systems. *Chinese Journal of Computers*, 31(12): 2090–2096, 2008.

[15] Gh. Păun. *Membrane Computing – An Introduction.* Berlin, Springer, 2002.

[16] Gh. Păun. Spiking neural P systems with astroyte-like control. *Journal of Universal Computer Science*, 13(11): 1701–1721, 2007.

[17] Gh. Păun. A quick overview of membrane computing with some details about spiking neural P systems. *Frontiers of Computer Science in China*, 1(1), 37–49, 2007.

[18] T. Song, L. Pan, and J. Wang. Normal forms for spiking neural P systems with anti-Spikes. Manuscript in preparation.

[19] J. Wang, H.J. Hoogeboom, and L. Pan. Spiking neural P systems with neuron division. In: M. Gheorghe, T. Hinze, Gh. Păun, G. Rozenberg, and A Salomaa (Eds.), *Membrane Computing, 11th International Conference. Lecture Notes in Computer Science* 6501, pages 361–376, Springer, 2010.

[20] J. Wang, H.J. Hoogeboom, L. Pan, Gh. Păun, and M.J. Pérez-Jiménez. Spiking neural P systems with weights. *Neural Computation*, 22, 2615–2646, 2010.

[21] X. Zhang, J. Wang, and L. Pan. A note on the generative power of axon P systems. *International Journal of Computers Communications and Control*, 4(1):92–98, 2009.