# Application of Formal Framework to Spiking Neural P System Variants: A Proposal

Ren Tristan A. de la Cruz

November 8, 2020

## 1 Introduction

*Membrane computing* is a field of computer science that studies models of computation known as *P systems*. *P systems* refer to a family of models of computation that are inspired by different biological processes. P systems use biological concepts like cells, cell membranes, neurons, tissues, etc. Rules (computating operations) in P systems are inspired by biological processes like chemical reactions in cells, ion transport between regions divided by membranes, membrane creation, division and dissolution, spiking of neurons, neurogenesis, and synaptogenesis. P systems are parallel and distributed models of computation. This means a P system can apply multiple rules at the same time and these rules maybe applied in different parts/components of the P system.

There is no single formal definition for the term 'P system' but there are certain characteristics that apply to most P systems. Most P systems use multiset of symbols as computing elements. One can think of these symbols as molecules or ions. The multiset of symbols are compartmentalized into regions and these regions are defined by membranes that enclose them. i.e. Region 1 is the space enclosed by membrane 1. This is the reason why the field is known as 'membrane' computing. Regions can be connected to each other. For example, in cell-like P system, membranes in the cell can be nested. If there are two membranes in a cell, membrane 0 and membrane 1, and membrane 1 is inside membrane 0 then the region enclosed by membrane 0 that is outside mebrane 1 is 'connected' to the region enclosed by membrane 1. In tissue-like P system, membranes represent cells and a cell enclosed one region. Cells (and hence regions) are connected to each other via channels. Tissue P systems form networks of cells. In general, the *configuration* of a P system refers to the network of cells/membranes with each cell/membrane containing a multiset of objects (the multiset can be empty). In some P systems, a cell/membrane has an internal state/status that is different from its content (multiset of symbols). For example, in some cell-like P system, a membrane has a state known as *polarity* or *charge* which can either be negative, positive, or neutral.

At the time of writing, there are probably hundreds of P system models. One reason for the diversity of P systems is the diversity of the types of rules used by the systems. There are many different types of rules and a P system model can use more than one type. A rule is an operation that transforms the configuration of a P system. One can look at a rule as having two components: (1) a set of conditions the configuration of a P system has to meet for the rule to be eligible for application, and (2) a set of actions (transformations) the rule will apply to the P system's configuration. Different types of rules have different sets of conditions and different actions since they are inspired by different biological processes. For example, one common type of rule is called an *evolution rule*. An *evolution rule* is associated with a region and it consumes a multiset of objects in that region then produces another multiset of objects. Evolution rules are inspired by chemical reaction inside the membranes of a cell. Another example of a rule type is the *communication rule*. In cell-like P systems, a communication rule is associated with a membrane and it facilitates a transfer or exchange of multisets of objects between the region inside the membrane and the region outside. Communication rules are inspired by the ion transfers between regions via ion channels on cell membranes. There are also types of rules that transform the network of cells/membranes itself and not just the contents of the cells/membranes. For example, there are rules that can create and delete channels between cells. There are also rules that can create and delete membrane/cells. These rules are inspired by processes like cell/membrane division, membrane dissolution, neuron creation, synapse creation and deletion, etc.

P systems are parallel models so it is possible for the systems to apply different rules at the same time and they can also apply a rule multiple times. For example, evolution rules are inspired by chemical reactions and it is possible for different chemical reactions to occur multiple times inside cell membranes. P systems have a set of criteria that specifies which combination of eligible rules are valid. This set of criteria is called the *derivation mode*. Aside from the types of rules used by P systems, different P systems can also use different derivation modes. Derivation modes can describe the type of parallelism the P system is working on. For example, the derivation mode known as *maximal parallelism* only allows combinations of eligible rule instances that are *maximal*. In a *maximal* combination of rule instances, one can not add any instance of any eligible rules. i.e The maximal combination of rules will consume enough symbols such that the remaining multiset of symbols does not allow any more rules to be applied. In a P system working in maximally parallel mode, any rule combinations of eligible rules that are not maximal are not allowed. Another common derivation mode is the *minimally parallel* mode. In P systems working in minimally parallel mode, cells/membranes with eligible rules can still apply rules in parallel but each cell/membrane can only apply a single eligible rule once. Derivation modes can also describe rule priorities. For example, in a P system with two types of rules, type $A$ and type $B$, one can specify that in the model if any type $A$ rule in a membrane is eligible then no (eligible) type $B$ rules can be applied. In the situation where there is an eligible type $A$ rule in a membrane, any rule combination that includes an eligible type $B$ rule in the same membrane is not valid.

When one combines the different types of rules and the different derivation modes, one can produce a significant number of diverse P system models. In order to help make sense of the different P system models a *formal framework* was introduced in 2007 by Freund et al. The idea behind the formal framework (for P systems) is to have a set of general concepts and constructs that can be used when analysing most (if not all) types of P systems. The first version of the formal framework can be found in [2] and it is a framework for static tissue-like P system. Similar to most P systems, a configuration in the framework is a network of cells that contain multisets of symbols. The framework has a single type of rule called an *interaction rule*. An *interaction rule* is a rule type that generalizes most rule types used by cell-like and tissue-like P systems. For example, most variants of communication rules and evolution rules are special cases of the more general interaction rule. Aside from having different forms, different types of rules can also have significantly different syntax. By using the interaction rule form, P system rules can be written to have the same general form and a common syntax. The interaction rule form makes comparing rules of different P systems easier. For example, two rule types from two different P systems may superficially appear different but when they are written as interaction rules it appears that their eligibility criteria and their actions are actually very similar. In the formal framework, effect of an interaction rule when applied to configuration is formally defined. This is not necessarily the case for other P systems and their rules. In some P system models, the effect of the rules on the configuration is described in a less formal manner (describe in plain English) which is prone to different interpretation. If one is to write a rule type from one P system model as an interaction rule, one has to formally define the meaning of the rule. The framework helps one clarify the meaning of the rule.

Aside from a general form of a rule, the framework in [2] also lists common derivation modes used by most P systems. The meaning of those different derivation modes are formally defined. In some P systems, derivation modes are less formally defined which again can lead to multiple conflicting interpretations. The framework lists other possible derivation modes that are rarely or not yet used by existing P system models.

The formal framework from [2] is limited to static P system. One can not represent a P system rule that changes the network of cells itself and not simply the content of the cells using the interaction rule in [2]. i.e. The rules that can create and delete cells and connections between cells can not be written as interaction rules. The second version of the formal framework [1] has a much more general form of a rule. This rule form can be used to write rules that change the structure of the network of cells. The rule form is much more complicated that the form of the interaction rules in [2]. A third version of the formal framework can be found in [4]. This version is actually an extension of the first version. The main differences between the first and the third version are: (1) the interaction rule in the third version allows multiset pattern checking and (2) the third version framework introduces the input and output constructs. In the first version of the framework, the criteria for eligibility of interaction rules are combination of presence of certain multisets in specified regions and absence of certain multisets in specified regions. In the third version, the criteria for eligibility of an interaction rule rely on *control languages* (the 'pattern'). For example, if certain multisets in specified regions are in the control languages of those regions (the multisets 'fits the pattern' for those regions), then the rule is eligible. Control languages and the input and output constructs are needed to represent P systems known as *spiking neural P systems* (SN P systems). SN P systems use a type of rule that checks for patterns. Such rules need the concept of a control language in order for them to be represented in the framework. SN

P systems are also usually used as transducers so it makes sense for the new framework to have input and output constructs.

The formal frameworks can be used to understand the functioning of some P system variants. This is done by translating concepts of a P system model to concepts in the framework. Since the concepts in the framework are formally defined, the process of translating P system concepts to the language of the framework forces one to clarify and formalize those concepts that may have been vaguely described in the P system model. This process is particularly helpful when analysing P system models with vague semantics. The process highlights the concepts (e.g. rule semantics) that may have different interpretations. The formal frameworks can also be used as a common language for comparing different P systems. By translating different P systems as models in the formal framework, the formal framework models for the different P systems will have the same general form and will use the same syntax. This makes the comparison of those models a significantly easier task. The formal framework can also be use to extend P system models with new features. Since the constructs in the formal framework can be seen as generalized versions of constructs in most P system models (i.e. formal framework's interaction rule is a generalized P system rule), one can add features available in the formal framework constructs as new features in the P system model. These of the formal framework can be found in [3].

## 2 Proposal

We want to do an exploratory study of the different SN P system variants using the formal frameworks.

SN P systems and similar variants are neural-like P systems. The rules of these models are inspired by the mechanism of a *spiking neuron*. In most SN P system variants, there is only a one type of symbol called a *spike*. The configuration of an SN P system is a network of neurons and each neuron contains a multiset of spikes. The main type of rule used by SN P system variants is called a *spiking rule*. A spiking rule looks at number of spikes in a certain neuron and if the number of spikes fits the pattern (usually described by a regular expression) specified in the rule then the neuron can send a spike to adjacent neurons in the network. The interaction rule in third version of the formal framework [4] was specifically extended to have control languages in order to be able to represent the spiking rule as interaction rules.

There are many SN P system variants that have features that are not available in the original SN P system model. For example, there is a variant with two types of objects known as the *spike* and the *anti-spike*. The variant is called SN P system *with anti-spikes*. The spiking rule of this variant needs to take into account the new type of object which means both its syntax and semantics are different from the original spiking rule. It also works in a slightly different derivation mode. There is a variant called SN P system *with astrocytes*. It has a new construct called *astrocyte*. An astrocyte can be connected to different channels and it monitors the spikes travelling in those channels. It is a threshold mechanism that allows travelling spikes to propagate if the total number of spikes travelling in the channels being monitored by the astrocyte passes a certain threshold. There is a variant called SN P system *with polarity.* Similar to some cell-like P systems, in SN P systems with polarity, neurons have charges that can either be positive, negative or neutral. The spiking rules in this model not only check the number of spikes in the neuron but also the polarity of the neuron. There are also SN P system variants that are exactly like the original SN P system except for the fact that they use a different derivation modes. For example, *sequential* SN P systems are SN P systems that only allows a single rule to be applied at a time. Another example is the *asynchronous* SN P system. This variant allows any combination of eligible rules (the combination does not have to be minimally parallel).

In [4], after defining and describing the constructs of the framework, the authors translated the SN P system model to a model of the framework. They wrote both the spiking rule and forgetting rule of the SN P system as interaction rules. They mentioned that the SN P system works in minimally parallel mode which has already been characterized as the derivation mode $min_1$ in [2]. Only original SN P system model (without delay) and SN P systems with extended rules (also without delay) have been fully translated as models of the framework. Some features of other SN P variants were translated by Verlan et al. in [4] and in some conference presentations (Conference on Membrane Computing) but not the full models.

Our intention for the study is to look at different SN P system variants and fully translate them to models of the framework. We want to fully translate at least two SN P variants. After translating that variants, we can them compare the framework models with each other. This can give us insights on how similar or how different those SN P system variants are to each other.

## References

[1] Rudolf Freund, Ignacio Pérez-Hurtado, Agustín Riscos-Núñez, and Sergey Verlan. A formalization of membrane systems with dynamically evolving structures. *International Journal of Computer Mathematics*, 90(4):801–815, 2013.

[2] Rudolf Freund and Sergey Verlan. A formal framework for static (tissue) p systems. In George Eleftherakis, Petros Kefalas, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, pages 271–284, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[3] Sergey Verlan. Using the formal framework for p systems. In Artiom Alhazov, Svetlana Cojocaru, Marian Gheorghe, Yurii Rogozhin, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, pages 56–79, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg.

[4] Sergey Verlan, Rudolf Freund, Artiom Alhazov, Sergiu Ivanov, and Linqiang Pan. A formal framework for spiking neural p systems. *Journal of Membrane Computing*, Oct 2020.