A Note on Spiking Neural P systems with Dynamic Structure versus Kernel P Systems (Extended Abstract)

Răzvan Vasile[1],
Marian Gheorghe[2],
Florentin Ipate[1],
Lakshmanan Kuppusamy[3], and
Ionuţ Mihai Niculescu[1]

[1]University of Bucharest, Faculty of Mathematics and Computer Science,
14 Academiei St, 010014 Bucharest, Romania.
[2]Department of Computer Science, University of Bradford,
West Yorkshire, Bradford BD7 1DP, UK.
[3]School of Computer Science & Engineering, VIT, Vellore - 632 014, India.

Abstract. In this note we consider spiking neural P systems with neuron division, budding and dissolution and spiking neural P systems with structural plasticity, and analyse various ways of mapping them into kernel P systems. Examples of different complexities illustrate the approach, making suggestions for an efficient mapping. Initial steps in applying formal methods in verifying these systems by using SPIN, from kPWorkbench, and Pro-B with Event-B, are also presented.

Keywords: Membrane Systems · Spiking Neural P Systems · Kernel P Systems · Formal Verification · Model Checking

1 Introduction

Membrane computing is a bio-inspired computing paradigm using models called *membrane systems* or *P systems*. These models are inspired by the structure and functionality of the living cell. The first model of this type has been introduced by Gh. Păun [14].

The original definition of a membrane (or P) system [14] consists of a *membrane structure* composed of several compartments, also called *membranes*, hierarchically embedded in the main membrane, called the *skin membrane*. Each compartment contains *objects* representing abstractions of the bio-chemical entities, from simple molecules to more complex DNA strands that appear in the living cell. Inside of each compartment there are also *evolution and communication rules*. The hierarchical structure, in the form of a tree, of such membrane systems, has then been replaced by an undirected graph, in the so called *tissue P systems*.

A first research monograph [15], a comprehensive handbook [16] and a survey paper [19] represent relevant contributions illustrating some of the most

significant achievements in membrane computing. Membrane computing is now a well-established nature inspired or unconventional computing research area. Some of the most recent research textbooks show the great interest in developing applications [4, 21].

Membrane computing area contains a plethora of models, called very often variants of membrane (or P) systems. One of such variants, called *Spiking Neural P system* (*SN P system*, for short), introduced in [7], is amongst the most popular, with a lot of variations.

So far, have been published a bibliography [13], survey papers on generic SN P systems and their applications [18, 10], a survey on learning aspects of SN P systems [3], as well as open problems and research topics in SN P systems [17].

Another type of P system, called *kernel P system* (*kP system*, for short), introduced in [6], aims to provide a modelling approach that combines in a coherent way features of existing P systems with new ones. Kernel P systems have been used for describing computer science problems, such as communication and synchronisation [5], or for modelling synthetic biology systems [8]. These models are specified in a domain specific language, called *kP-Lingua*, allowing models to be simulated with a software framework, called kPWorkbench [1]. The capabilities of the tool are presented in [9].

In this note we investigate the mapping of various SN P systems with dynamic structure (i.e., with plasticity [2, 11], neuron division and budding [12], neuron division and dissolution [22]) into kP systems and provide some initial steps in analysing these systems with formal methods, such as model checkers.

2 Main Contributions

This work aims to: (i) elaborate a sound methodology for mapping SN P systems with dynamic structure into equivalent kP systems and (ii) on the account of the kP system formalism, demonstrate some properties of the kP system (and SN P system) model, using model checkers, such as Pro-B and SPIN.

We present here a first SN P system with dynamic structure that was explored in [2], as an SN P system with plasticity – see Figure 1.

This SN P system consists of five neurons. Neurons σ_1 and σ_3 contain initially two spikes and one spike, respectively and the rest have no spikes. The rules of σ_1 are depicted in Figure 1 and the other neurons, i.e., $\sigma_2, \sigma_3, \sigma_{A_1}, \sigma_{A_2}$, contain only one rule: $a \to a$, which was omitted in writing.

The application of the rules in σ_1 is deterministic. The nondeterminism in the example occurs in the process of selecting which synapses to create or delete. As such, two synapses (1,2) or (1,3) can be created in neuron σ_1. If the synapse (1,3) is created, then σ_1 sends one spike to σ_3; in the next step, the synapse (1,3) is deleted and σ_3 sends out to the environment the spike received from σ_1. If, however, the synapse (1,2) is created, then the behaviour of the system is similar, but in this case, neuron σ_2 sends one spike to each of the auxiliary neurons σ_{A_1} and σ_{A_2}. As long as σ_1 creates synapse (1,2), the system keeps receiving two spikes in a loop.

This example has been mapped into three kernel P system models, and all of them have been simulated into *kP Lingua*, using kPWorkbench. One such kP model, translated into *kP Lingua*, can be found on GitHub [20], where the results of some simulations are also provided.

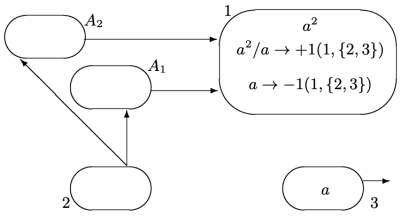

Fig. 1: SN P system with dynamic structure from [2].

The first kP system model is a one-to-one mapping of the example from [2], albeit, with minor adjustments to fit the formalism of *kernel P systems*. In this model, links between compartments are dynamically created and destroyed.

Other two *kP system* models are optimised versions of the first one. One of them includes links that are dynamically created and destroyed, and the other one doesn't.

An equivalent Event-B model has been created for each of these three *kP system* models. Although Event-B has been prior used to model some kP systems, it is the first time when *kP systems* are translated into Event-B.

For each Event-B model, given two general sets $SYMBOLS$ and $TYPES$, two invariants have been created:

- **inv1:** $MULTISETS \subseteq TYPES \times \mathbb{N} \times SYMBOLS \times \mathbb{N}$;
- **inv2:** $LINKS \subseteq TYPES \times \mathbb{N}$.

Using the two invariants, one can model into Event-B any *kP system* in a very natural way: each rule from the *kP system* is an event in the Event-B model, and any guard from the *kP system* is an Event-B guard.

We have considered the first *kP system* mentioned above for proving several properties, using the Pro-B model checker, based on the Event-B model:

1. Eventually, in the future, both neurons σ_1 and σ_3 will have a spike.
2. If neuron σ_1 has two spikes, then eventually (F) in the future the neuron σ_3 might have a spike.

The kPWorkbench framework has the integrated component of formal model checking, using SPIN. All this is automated, as part of the overall functionality of kPWorkbench. We have proved these properties:

When neuron σ_1 eventually spikes to neuron σ_3, in the next step, neuron σ_3 will contain one spike.
eventually N1.n3 = 1 implies (next N3.a = 1)
When neuron σ_{A1} eventually spikes to neuron σ_1, then neuron σ_1 will eventually spikes to neuron σ_3.
eventually NA1.a = 1 implies (eventually N1.n3 = 1 and (next N3.a = 1))
After receiving the initial spike, the environment will eventually receive the second spike.
eventually environment.a = 1 implies (eventually environment.a = 2)

Table 1: Model properties

3 Conclusions and Future Developments

The ongoing research on the topics of this note is looking at defining the methodology of mapping SN P systems with dynamic structure, namely SN P systems with plasticity and neuron division, budding and dissolution, by considering different algorithms and specific cases and to adequately associate Event-B models, comparing them with SPIN ones, in order to prove a broad palette of properties.

References

1. Bakir, M.E., Ipate, F., Konur, S., Mierlă, L., Niculescu, I.-M.: Extended simulation and verification platform for kernel P systems. In: Gheorghe M. et al (ed.) 15^{th} Int. Conference on Membrane Computing, LNCS 8961, pp. 158–178 (2014)
2. Cabarle, F.G.C., Adorna, H.N., Pérez-Jiménez, M.J., Song, T.: Spiking neural P systems with structural plasticity. Neural Computing and Applications **26**(8), 1905–1917 (2015)
3. Chen, Y., Chen, Y., Zhang, G., Paul, P., Wu, T., Zhang, X., Rong, H., Ma, X.: A survey of learning spiking neural P systems and a novel instance. International Journal of Unconventional Computing **16**(2–3), 173–200 (2021)
4. Frisco, P., Gheorghe, M., Pérez-Jiménez, M.J. (eds.): Applications of Membrane Computing in Systems and Synthetic Biology. Springer, Verlag (2014)
5. Gheorghe, M., Ceterchi, R., Ipate, F., Konur, S., Lefticaru, R.: Kernel P systems: from modelling to verification and testing. Theoretical Computer Science **724**, 45–60 (2018). URL `http://hdl.handle.net/10454/11720`
6. Gheorghe, M., Ipate, F., Dragomir, C., Mierlă, L., Valencia-Cabrera, L., García-Quismondo, M., Pérez-Jiménez, M.J.: Kernel P Systems - Version I. 11^{th} Brainstorming Week on Membrane Computing pp. 97–124 (2013). URL `http://www.gcn.us.es/files/11bwmc/097_gheorghe_ipate.pdf`
7. Ionescu, M., Păun, Gh., Yokomori, T.: Spiking neural P systems. Fundamenta Informaticae **71**(2–3), 279–308 (2006)
8. Konur, S., Gheorghe, M., Dragomir, C., Ipate, F., Krasnogor, N.: Conventional verification for unconventional computing: a genetic XOR gate example. Fundamenta Informaticae **134**(1–2), 97–110 (2014)
9. Konur, S., Mierlă, L., Ipate, F., Gheorghe, M.: kPWorkbench: A software suit for membrane systems. SoftwareX **11**, 100407 (2020)
10. Leporati, A., Mauri, G., Zandron, C.: Spiking neural P systems: main ideas and results. Natural Computing **21**(4), 629–649 (2022). https://doi.org/https://doi.org/10.1007/s11047-022-09917-y

11. Macababayao, I.C.H., Cabarle, F.G.C., de la Cruz, R.T., Zeng, X.: Normal forms for spiking neural P systems and some of its variants. Information Sciences **595**, 344–363 (2022)
12. Pan, L., Păun, Gh., Pérez-Jiménez: Spiking neural P systems with neuron division and budding. Science China Information Sciences **54**, 1596–1607 (2011)
13. Pan, L., Wu, T., Zhang, Z.: A bibliography of spiking neural P systems. Bulletin of the International Membrane Computing Society (I M C S) **1**(1), 63–78 (2016)
14. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences **61**(1), 108–143 (2000). https://doi.org/10.1006/jcss.1999.1693. URL `https://doi.org/10.1006/jcss.1999.1693`
15. Păun, Gh.: Membrane Computing - An Introduction. Springer, Verlag (2002)
16. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
17. Păun, Gh., Wu, T., Zhang, Z.: Open problems, research topics, recent results on numerical and spiking neural P systems (The 'Curtea de Argeş 2015 series'). In: Proceedings of Fourteenth Brainstorming Week on Membrane Computing, pp. 285–300. Sevilla, Spain: Fenix Editora (2016)
18. Rong, H., Wu, T., Pan, L., Zhang, G.: Spiking neural P systems: Theoretical results and applications. In: Graciani, C. et al. (ed.) Enjoying Natural Computing, LNCS 11270, pp. 256–268 (2018)
19. Song, B., Li, K., Orellana-Martín, D., Pérez-Jiménez, M.J., Hurtado, I.P.: A survey of nature-inspired computing: Membrane computing. ACM Computing Surveys **54**(1), 629–649 (2021). https://doi.org/https://doi.org/10.1145/3431234
20. Vasile, R.: (2024). URL `https://github.com/Razvan-V/SNP-kP/tree/main/Example_2`
21. Zhang, G., Pérez-Jiménez, M.J., Gheorghe, M.: Real-life Applications with Membrane Computing. Springer, Verlag (2017)
22. Zhao, Y., Liu, X., Wang, W.: Spiking neural P systems with neuron division and dissolution. PLOS ONE **11**(9), e0162882 (2016)