Kernel P systems and connections with membrane systems (Extended Abstract)

Marian Gheorghe[1]

[1]School of Computing and Engineering, University of Bradford, West Yorkshire, Bradford BD7 1DP, UK

Abstract

The relationships between a more general class of membrane systems, called kernel P systems, and other variants of membrane systems are presented. The process of translating the main types of spiking P systems into kernel P systems are analysed with respect to certain complexity metrics. Formal verification, through model checking, and testing procedures are discussed for the models obtained from these mappings.

Keywords

Membrane computing, kernel P systems, spiking neural P systems, computational power, formal verification (model checking) and testing

Membrane computing is a natural computing paradigm inspired by the structure and bio-chemical interactions of the living cells. The key models are called *membrane systems* or *P systems*. A standard (basic) membrane system model consists of three main elements: (i) a set of components, called *membranes* or *regions*, organised in a hierarchical structure and included in an outer membrane, called the *skin membrane*; each region contains (ii) *multisets of objects* and (iii) *evolution and communication rules* acting on the local objects, transforming or sending them to the regions from its vicinity. The main membrane-based structure is inspired by the eukaryotic cell's structure containing a nucleus and specialised membrane-bound organelles such as mitochondria, endoplasmic reticulum, Golgi apparatus. The objects encode either simple elements (hydrogen, oxygen, nitrogen, carbon, sulfur) or bio-chemical compounds (carbohydrates, lipids, proteins, and nucleic acids) and the rules stand for various interactions (chemical reactions, more complex transcription and translation, cell division and cell death processes).

A membrane system is a theoretical (mathematical) model meant to provide a highly parallel and distributed computing device with interactions amongst objects running in parallel both at the system level and in each compartment. The computational power of many variants of this model is equivalent to that of a Turing machine and, due to their highly parallel behaviour, time efficient solutions to many difficult problems are provided.

The field has developed very fast on various research directions, with many classes of membrane systems (P systems) being defined and investigated. These may be classified as *cell-like*, *tissue-like* and *neural-like* P systems, corresponding to various system structures considered for the model. The most important membrane computing variants, as well as key theoretical developments, connections with other computational models and the most relevant applications, have been presented in a handbook [1] and later on summarised based on various criteria in a survey paper [2].

A special type of P systems, called *kernel P systems* (*kP systems*, for short), has been introduced in [3] in order to capture in a unified manner features of existing membrane computing models together with new concepts, such as types and compartments instantiated from them, guarded rules using Boolean conditions, creation and destruction of links between compartments, user defined execution strategy for each component type. All these make the kP system model more amenable for describing different problems occurring in various areas, from specific computer science topics, such as communication and synchronisation [4], to applications in synthetic biology [5]. kP systems have been conceived as a membrane computing model allowing to specify problems, verify the models correctness and test

ITAT'25: Information Technologies – Aplications and Theory, September 26–30, 2025, Telgárt, Slovakia
m.gheorghe@bradford.ac.uk (M. Gheorghe)
https://www.bradford.ac.uk/staff/mgheorghe// (M. Gheorghe)
0000-0002-2409-4959 (M. Gheorghe)

the implementations. They are supported by an expressive domain specific formal language allowing the models to be simulated with a software framework, called kPWorkbench, which also includes a verification component. A detailed presentation of this tool has been made in [6]. The modelling, simulation, verification and testing aspects of kP systems have been presented in [4].

A class of neural-like P systems, called *Spiking Neural P systems* (*SN P systems*, for short), has been introduced in [7], inspired by the neuro-physiological behaviour of neurons (in the brain) sending electrical impulses along axons to other neurons. This class of P systems can be regarded as a special type of spiking neural networks (*SNNs*), a special class of artificial neural networks mimicking more closely the biological neurons. Similar to SNNs, SN P systems use discrete spikes as activation mechanisms. SN P systems have been initally introduced as computing devices and then learning capabilities have been added to them and used in various applications. A multitude of SN P system models have been considered and investigated with features such as: anti-spikes and inhibitory synapses, inhibitory rules, astrocytes, polarizations, with multiple channels, with weights and potential (threshold) and many others. A survey of the main classes of SN P systems is presented in [8]. Theoretical results and applications of SN P systems are discussed in a 2024 research textbook dedicated to this subject [9]. For several of the above mentioned SN P systems have been considered different strategies for using the rules as well as ways of encoding the results of the computation – the key topics in this respect have been presented in [8].

In this work are first presented some relationships between, on the one hand, kP systems and, on the other hand, neural-like P systems and membrane systems with active membranes. The main focus is on describing how the classes of standard and extended SN P systems with and with no delay, with colored spikes, multiple channels, anti-spikes, polarizations, weights on synapses, potential and threshold. are mapped into kP systems exhibiting the same behaviour. Two types of encoding the computation results and various strategies of using the rules that are defined and investigated for SN P systems are studied in the context of kP systems as well. A logical gate example illustrates how different SN P systems and their kP system counterparts can model the problem. Finally, a short presentation on how verification and testing methods developed for kP systems can be used to formally verify the SN P system model of the logical gate example and test its implementation.

Algorithms for translating each of the SN P systems investigated into corresponding kP systems are provided and metrics to assess the impact of various features of the classes of SN P systems considered on the complexity of their kP system representations are presented. Moreover, the instrumentation provided by the kPWorkbench environment facilitates the verification of the model correctness, through model checking methods, and the automated testing of the implementations based on the initial SN P systems.

Some future research developments based on the work reported here are: (i) investigate relationships between other classes of SN P systems and kP systems; (ii) extend the set of features of kP systems such that they provide better mappings for the SN P systems, with respect to the complexity metrics introduced; (iii) identify more complex examples or case studies based on SN P systems and apply formal verification and testing procedures to their corresponding kP systems, tracing the results back to the original models.

Acknowledgements

This work reported here has been partially supported by the Royal Society grant IES\R3\213176, 2022-2025.

Declaration on Generative AI

The author has not employed any Generative AI tools.

References

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

[2] B. Song, K. Li, D. Orellana-Martín, M. J. Pérez-Jiménez, I. P. Hurtado, A survey of nature-inspired computing: Membrane computing, Communications of the ACM 54 (2021) 629–649.

[3] M. Gheorghe, F. Ipate, C. Dragomir, L. Mierlă, L. Valencia-Cabrera, M. García-Quismondo, M. J. Pérez-Jiménez, Kernel P systems - Version I, in: 11^{th} Brainstorming Week on Membrane Computing, Fénix Editora, 2013, pp. 97–124.

[4] M. Gheorghe, R. Ceterchi, F. Ipate, S. Konur, R. Lefticaru, Kernel P systems: from modelling to verification and testing, Theoretical Computer Science 724 (2018) 45–60.

[5] S. Konur, M. Gheorghe, C. Dragomir, L. Mierlă, F. Ipate, N. Krasnogor, Qualitative and quantitative analysis of systems and synthetic biology constructs using P systems, ACS Synthetic Biology 4 (2015) 83–92.

[6] S. Konur, L. Mierlă, F. Ipate, , M. Gheorghe, kPWorkbench: A software suit for membrane systems, SoftwareX 11 (2020) 100407.

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

[8] A. Leporati, G. Mauri, C. Zandron, Spiking neural P systems: main ideas and results, Natural Computing 21 (2022) 629–649.

[9] G. Zhang, S. Verlan, T. Wu, F. G. C. Cabarle, J. Xue, D. Orellan-Martín, J. Dong, L. Valencia-Cabrera, M. J. Pérez-Jiménez, Spiking Neural P Systems - Theory, Applications and Implementations, Springer, Verlag, 2024.