Enhancing P Systems for Complex Biological Simulations

A.A. Elsayed[1,3], R. Ceprián[1,2], A. Hafez[1], C. Llorens[1], and J.M. Sempere[3⋆]

[1] Biotechvana, Carrer del Catedràtic Agustín Escardino, 9, 46980, Paterna, Spain
[2] Institute for Integrative Systems Biology (I2SysBio), Universitat de València - CSIC, Carrer del Catedràtic Agustín Escardino, 9, 46980, Paterna, Spain
[3] VRAIN, Universitat Politècnica de València, Camino de Vera s/n, 46022, Valencia, Spain
ayoya.ali@biotechvana.com raquel.ceprian@biotechvana.com
ahmed.hafez@biotechvana.com carlos.llorens@biotechvana.com
jsempere@dsic.upv.es

Abstract. Membrane computing, specifically through P systems, has significantly advanced computational biology, by providing sophisticated models to address the computational challenges posed by the dynamic nature of biological systems. This work highlights the limitations encountered in existing P system frameworks, particularly in modeling real biological life systems. Addressing these limitations, we introduce innovative rules aimed at enhancing the representational fidelity of P systems for biological computations. In addition, we introduce two visualization tools that help the debugging and validation of P systems, by showing the relationships between objects, membranes and rules.

Keywords: membrane computing · P system rules · systems biology

1 Introduction

In the realm of computational biology, membrane computing, particularly through P systems, has emerged as a very suitable tool for modeling and simulating complex biological processes [1–4]. Membrane computing was initiated by Gheorghe Păun in 1998 [5, 6] as a discipline where P systems, inherently inspired by the structure and functioning of living cells, offer a unique approach to computational thinking, deeply rooted in cellular dynamics. These systems are traditionally defined by a formal set of rules dictating the interactions and transformations within bio-inspired models. While potent, these rules frequently require evolution to match the growing complexity and nuanced needs of biological simulations.

During our work in progress to model the human immune system, we encountered limitations within the existing P system rules framework, underscoring the necessity for innovation. While P systems lay a solid foundation, the intricacies

⋆ Corresponding author

of the immune system demanded a more customized approach. This realization propelled the development of new rules within the P system paradigm, striving to bridge the gap between theoretical models and the practical requisites of biological computations.

The genesis of these new rules was inspired by the objective to model the human immune system's response to vaccines and viral infections. In an effort to accurately represent biological processes, we modeled organs and cells as membranes within the P system, while key entities like *interleukins* and *immunoglobulins* [7–10] were portrayed as objects. These rules facilitate the representation of interactions between membranes and objects, allowing for movements such as object migration, duplication, and replication, thus mimicking biological processes. For instance, in the nervous system (NS), and more specifically the central nervous system (CNS), interleukins present before and after vaccination or infection are simulated as objects in the P system. These interleukins, originating from various organs or generated in response to other object interactions, reflect the biological activities occurring in the body.

Our contributions include the formulation of a new kind of rules to manage empty membranes, that is a critical aspect overlooked in traditional frameworks. In addition, we detail the augmentation of P system rules, by introducing different probabilistic parameters in the rules, to enable a more nuanced simulation of biological interactions. In addition, we show computer tools for visualizing the membrane structure and the rules dependencies of P systems.

The structure of this work is the following: In Section 2, we provide some background on membrane computing and we define formally a P system with active membranes. In Section 3, our proposal for a new kind of rules to check the emptyness of the system regions, and some aspects of probabilistic rules are described. In Section 4, we discuss some aspects of the implementations and results of the previous proposals, and we show computer tools that we have developed for the visualizing of the membrane regions and the dependencies between rules and regions. Finally, we provide some conclusions on our proposal, and we show our current work in progress related to it.

2 Basic definitions and background

By advancing the P system rule set and introducing a mechanism to visualize these computational models, this work seeks to narrow the gap between theoretical constructs and their practical application in simulating biological systems. Our objectives are threefold: to innovate new rules and definitions for modeling cellular dynamics, to enhance the computational capabilities of P systems for complex biological simulations, and to develop visualization tools for membrane computing, thereby providing a comprehensive platform for the exploration and understanding of biological processes through membrane computing.

The intersection of computer science and biology has led to significant advancements in understanding complex biological systems. In particular, membrane computing, introduced by Gheorghe Păun in 1998 [5, 6], stands out as a

groundbreaking approach inspired by the architecture of living cells. This field has evolved rapidly, leading to the development of P system models that offer a unique lens through which to view computational processes akin to those within eukaryotic cell membranes. These models, while powerful, continually adapt to capture the intricate dynamics of biological simulations more accurately. Bioinformatics [11] has experienced rapid growth, marked by the genomics and proteomics eras, revolutionizing the management and analysis of biological data. This discipline leverages computational tools to handle vast datasets, compare sequences, predict molecular structures, and model biological systems among other tasks, pushing the boundaries of how we study life at a molecular level.

Natural computing [12, 13] has given rise to models inspired by biological phenomena, encompassing fields like evolutionary algorithms, artificial neural networks, and molecular computing. These models draw from natural selection, brain function, and molecular interactions, respectively, to solve complex problems in innovative ways. DNA Computing emerged as a particularly active area within molecular computing, notably with Leonard Adleman's seminal experiment in 1994 [14]. This approach exploits the parallelism inherent in biochemical processes by using DNA to perform computations, which presents advantages in energy efficiency and data storage capacity. Membrane computing abstracts from the cell structure and internal processes, envisioning each compartment as part of a computational unit. Introduced by Păun, P systems formalize this concept, employing a membrane structure to represent the hierarchical organization of cells and using objects and evolution rules to simulate biological reactions and interactions. The computation within P systems unfolds through the nondeterministic and parallel application of these rules, demonstrating remarkable computational capabilities.

We provide a basic definitions on P systems to describe the new rules and behaviors that will be described later from [15] as follows.

Definition 1. *A P system with active membranes of degree $m \geq 1$ is defined by the tuple $\Pi = (O, H, \mu, w_1, w_2, \cdots, w_m, R, i_0)$ where*

1. *O is the alphabet of objects*
2. *H is the alphabet of labels for membranes*
3. *μ is the initial membrane structure, of degree m, with all membranes labeled with elements of H and no polarizations. A membrane with label h is represented as $[\ ]_h$*
4. *$w, w_2, \cdots, w_n$ are strings over O specifying the multiset of objects present in the compartment of μ*
5. *R is a finite set of rules of the following types*
   (a) $[v \to w]_h$ with $v, w \in O^*$ (evolution rules)
   (b) $v[\ ]_h \to [w]_h$ with $v, w \in O^*$ ('in' communication rules)
   (c) $[v]_h \to w[\ ]_h$ with $v, w \in O^*$ ('out' communication rules)
   (d) $[v]_h \to [\ [w]_j]_h$ with $v, w \in O^*$ (membrane creation with object evolution)
6. *$i_0 \in \{0, \cdots, m\}$ indicates the region where the result of a computation is obtained (0 represents the environment).*

The rules of the P system are applied in a non-deterministic maximally parallel manner, and the computation of the system finishes whenever no rule can be applied. A configuration of the system at an instant t during a computation is defined by the membrane structure μ_t and the multiset of objects at every compartment in μ_t.

3 New rules for systems biology

In this section we are going to introduce a new type of rules to model biological systems using P systems. In addition, we are going to clarify some aspects of the existing rules in P systems when working with populations of objects that represent biological entities.

A new kind of rules to manage empty membranes

A key innovation in this work addresses the scenario of an *empty* cell, a membrane devoid of objects which, in biological terms, would typically die (if it is a living cell), or it should be dissolved (if it is an organelle or a vesicle). In the standard P system, a rule exists where if a particular object is present, the membrane dissolves, and the object moves to the parent membrane. However, our approach is focused on what happens if a membrane is empty, representing a cell in the body with no functional components. Scientifically, such a cell would not be viable. To address this, we introduced the phi symbol (Φ), representing the absence of objects in the referred region.

For example, an empty membrane dissolves can be represented as

$$[\Phi]_X \rightarrow \partial$$

Observe that the symbol ∂ is used to preserve the criterion that every rule has a left part and a right part, and it has no object meaning.

This rule not only aligns with biological understandings of cellular behavior but also significantly enhances the P system capacity to simulate these processes more accurately.

The Φ symbol can also be used in conjunction with other objects, in those cases it represents an *only if* condition, for example the following rule :

$$[\Phi, v]_X \rightarrow \partial$$

This rule only dissolves membrane X if it only contains objects represented by v and nothing else. This type of rules is very useful in systems biology, for example when modeling the cellular destruction caused by viruses once they have replicated within a cell.

Enhancing P systems for complex biological simulations

In a formal definition of P system a rule consist of input and output. Herein, we associate a set of properties to each rule. The properties include descriptive annotations (e.g., rule name), probabilities, priority and execution time.

Basic rules format consist for two major parts input and output

$$input \rightarrow output \quad (prop\_key = prop\_value)$$

For example, in the rule $[(vai)^2]_{SK} \rightarrow [(vai)^2, QC]_{SK}$ two objects vai inside the membrane SK produce a new object QC. Hence, the *input* establishes the required conditions to apply the rule while the *output* defines the expected result from applying the rules.

For any rule in the P system, we can define properties (*prop_key*). We will consider the following properties.

1. *name*
   The *name* is used to distinguish between similar rules that have the same input with different outputs.

2. *time*
   It is expected that every rule requires some time to produce the output. Time property can be specified with time units (e.g., seconds, minutes or hours). According to the time units, universal and local clocks shoud be introduced in the P system to trigger the different events for producing the outputs.
   For example, the following rule stats that it will take about two hours to be executed.

   $$[(vai)^2]_{SK} \rightarrow [(vai)^2, QC]_{SK} \quad (time = 2h)$$

   Timed rules will behave differently than other rules. If timed rules are used to simulate some biological aspect, when any system simulator encounters this type of rules, it consumes the required input objects, dissolves the established membranes (if any), and it stages the output to be produced after the required time. Time is translated to the equivalent number of iterations according the to the configured local or universal clocks of the system. Ideally the time unit of an iteration should be at least equivalent to the smallest time unit of any given rule.

3. *probability*
   The use of P systems to model biological phenomena involves giving them a probabilistic or stochastic flavor. Hence, the rules should stat a probability that determines whether it will execute under the right conditions. In a non-probabilistic manner, if the input conditions are met (meaning the objects or membranes specified in the rule are present), the rule is applied provided that there is no other competing rule. However, it is not a given that the rule will be executed every time those conditions are met. This is where

probability comes into play, adding a layer of realism to the simulations by recognizing that biological processes do not always follow a simple and deterministic path.

In the following section, we discuss how the system can define the probabilities so that any simulator obtains realistic results of biological phenomena.

Managing the probabilities

We categorize the randomness execution of each rule into three parts that take into account different characteristics of biological phenomena. The probabilistic approach will be drived by three components (IPP, RPI, RP) that we explain as follows.

1. *Input Population Percentage* (IPP)
   This is a crucial aspect of the rule enhancements for P systems to simulate biological systems. IPP specifically addresses the variability in the number of objects and membranes contained within a single membrane, which can range from a few to millions. Rules in P systems apply uniformly to all objects or membranes within a given membrane if they meet the rule conditions, such as possessing a specific object or being of a certain type. This universal application may not always align with the desired simulation outcomes, particularly in complex biological processes where interactions often occur on a probabilistic rather than a deterministic basis.
   To accommodate this variability and introduce a level of selective interaction, IPP is defined into the rules. IPP allows for the specification of a percentage that determines how many of the eligible objects or membranes within the parent membrane are affected by the rule. For instance, if a rule is meant to simulate a biological reaction that only occurs in a fraction of the cell population, IPP can be set to reflect this proportion, ensuring that the rule is applied selectively rather than universally. This approach significantly enhances the realism and accuracy of the P system simulations, allowing for a more nuanced representation of biological processes where not all interactions occur universally across all entities.

   Example 1.
   The rule $r2.0$ is applied to only 20% of the QC objects that are in the membrane SK. Hence, 20% of the QC objects disappear from the SK membrane, while 80% of the QC objects will remain unchanged in the membrane SK, according to this rule.

$$[QC]_{SK} \rightarrow []_{SK} \quad (\text{IPP} = 0.2, name = r2.0)$$

   □

2. *Rule Priority Index* (RPI)
   This concept addresses scenarios where multiple rules meet the conditions for application within the same membrane, potentially leading to different

outcomes based on the same set of inputs. RPI is particularly vital when rules are designed to simulate the transport of objects and membranes to other locations within the system but need to be executed in a controlled and sequential manner.
In our study, consider a membrane containing various objects and sub-membranes, where the objective is to distribute these elements to other membranes but not simultaneously or indiscriminately. A challenge arises when there is a risk of disproportionately sending objects or membranes to one location while neglecting others, or sending them to an unintended destination first. RPI is employed to mitigate such issues by assigning a priority level to each rule, ensuring that the most critical actions are executed first. This prioritization is also crucial when different quantities of an object within a membrane imply different biological conditions. For instance, a higher concentration of a specific biomarker might indicate a pathological state, whereas a lower concentration could be considered normal. If rules designed to interpret these concentrations are in conflict, RPI ensures that the correct interpretation is applied, avoiding potentially dangerous oversights. The utility of RPI is resolve the conflict between the same rules of the same input. In our proposal, the lowest RPI value means the highest priority.

Example 2. In the following rule $r1$, the presence of specific quantities of interleukins (IL objects) and tumor necrosis factors (TNF objects) within the DS membrane triggers the addition of the object $serv$, denoting a sharp pain response. The assigned RPI of 1 indicates that his rule takes precedence over the rest of the rules, ensuring that it is applied first to accurately represent severe biological conditions.

$$[(IL)^{50}((TNF)^{14}]_{DS} \rightarrow [(serv)(IL)^{160}(TNF)^{14}]_{DS} \quad (\text{RPI} = 1, name = r1)$$

The rule $r2$ introduces $assy$ object to the output, symbolizing a normal response based on the object concentrations. The rule with an RPI of 2 is firstly applied, provided that there is no rule with higher priority (i.e. RPI=1) over others while it is not important, there's another rule with higher precedence.

$$[(IL)^{40}(TNF)^{160}]_{DS} \rightarrow [(assy)(IL)^{68}(TNF)^{320}]_{DS} \quad (\text{RPI} = 2, name = r2)$$

□

3. *Rule Probability* (RP)
This associate a probability to the rule itself, hence modeling the randomness of executing the rule, if satisfy the input conditions. The utility of RP together with RPI will help to make some difference between the samples that share the same scenario as we mentioned above.

Example 3.
The following rule $r1.1$ will be applied 90 percent of times assuming it can be applied.

$$[(vai)^2]_{SK} \rightarrow [(vai)^2(QC)^2]_{SK} \quad (IPP = 0.8, RP = 0.9, name = r1.1)$$

□

Observe that IPP and RP control two independent events, first RP determines if the rule can be applied or not and, when the rule is applied, it is applied only to the IPP percentage of the objects. In the Example 3, only 90 % of times rule $r.1.1$ will be applied. whenever it be applicable (RP=0.9). Every time that the rule $r1.1$ is applied, it will only be applied over the to the 80 % of objects (IPP=0.8).

4 Implementation and results

In our approach to membrane computing, we leverage P systems to simulate biological processes by establishing a comprehensive framework for defining transformation rules. We implemented these rules within a Python-based simulator, crafted to apply them and predict outcomes from the specified interactions. This simulator is operated via a command-line interface (CLI), with usage showed as follows:

```
python ./src/sim.py ./scenario/target_scenario.ini
--max_iter 10 --n_samples 100
--save_tracking --write_yaml
```

This command triggers the script `sim.py` located within the `SRC` folder. The script is configurable via several parameters:

- The scenario file (`target_scenario.ini`) encapsulates the rules to be applied.
- `--max_iter` sets the number of iterations for the simulation.
- `--n_samples` defines the sample size, simulating a specified number of individuals.
- `--save_tracking` toggles the logging feature, which records the status of each iteration, detailing rule applications, membrane changes, and object transformations. This flag is set to `true` by including it in the command; its absence implies a `false` value.
- `--write_yaml` decides whether to output the simulation results in YAML format, providing a detailed iteration-by-iteration account.

The command line thus encapsulates both the input—in the form of our meticulously crafted rules—and the output, comprising various data representations. This includes CSV files for statistical analysis, YAML files for a comprehensive iteration overview, and HTML files for visualizing the system both during and after the application of rules, offering insights into the simulation end state.

To implement RPI effectively, we assign a numerical priority value to each rule, ranging from 1 (highest priority) to the total number of rules that require prioritization within the same context. This system allows us to manage rules that affect the same membrane and objects but yield different outputs, ensuring that the sequence of rule application accurately reflects the intended biological or processual logic.

4.1 Rules visualization

Complex rule-based P system interactions can be really difficult to diagnose. Visualize tools can facilitate designing such complex system. We have developed network diagrams that can be used for diagnostic, identify orphan rules, detecting unreachable simulations paths inside the system and enhancing our understanding and analysis of P system behaviors.

In Fig. 1, we show a network diagram obtained by a visualization tool that translates the rules and object dependencies into nodes and edges. In such network diagrams, blue triangle nodes are objects, green dots represent required input membranes for a given rule (e.g. rules that require the presence of some membranes, but do not require the inclusion of any objects in such membranes), and red square nodes are rules. Directed edges represent inputs and outputs of the rules. Object names use a full name convention in the form *parent_membrane.object_name* to distinguish them as there might be same objects in different membranes. For example $SK.vai$ will be the object name vai that belong the membrane SK and $Nesk.vai$ is the objects vai belong to the membrane $Nesk$. The rules are represented in red square nodes, the red arrows represent the rule input (left hand-side) and the blue arrows represent the output (right hand-side) as depicted in Fig. 2.

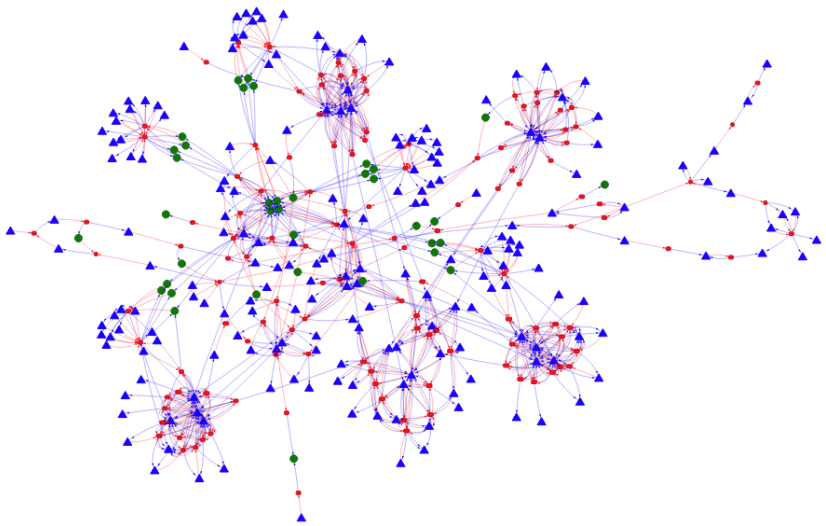

Fig. 1. A network diagram to show P system objects, membranes and rules dependencies.

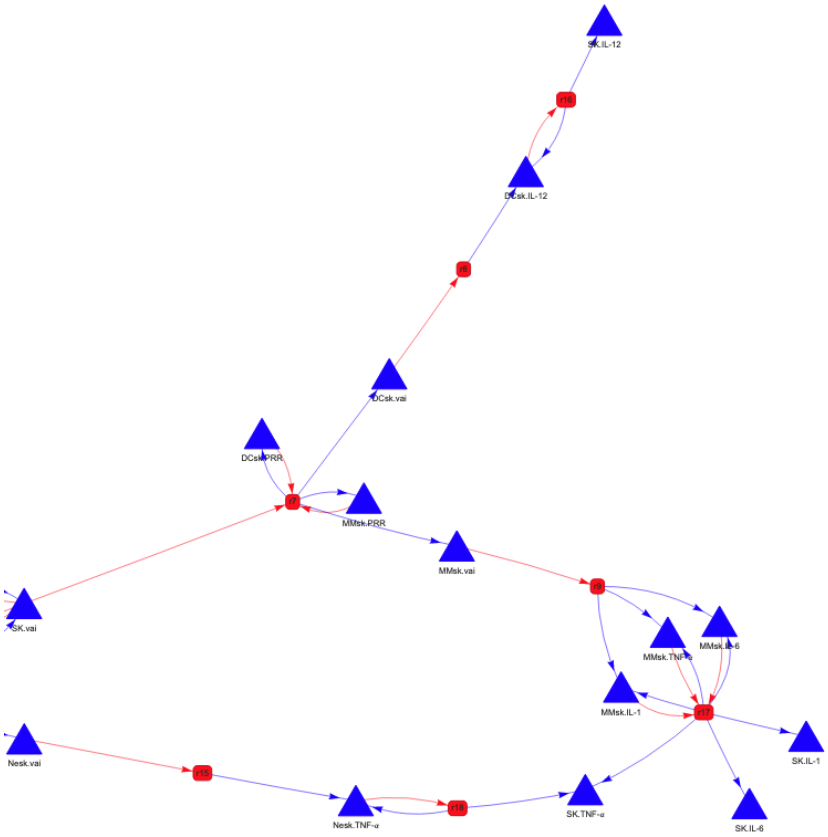

Fig. 2. A subnetwork of the Fig. 1 to highligh rule inputs and outputs

4.2 Membranes structure visualization

Visualizing the membranes structure at different time during the P system computation can be a useful diagnostic tool of any simulator behaviour. We have developed a computer tool that shows the membranes structure of the P system at a given time during a computation. In Fig. 3, we show a diagram obtained by our computer tool that contains a hierarchical and nested visualization of the membranes. Fig. 3 depict at each membrane level all its nested objects and membranes. This type of visualization can be also written by the simulation at each iteration as diagnostic and tracking the behaviour of the membranes during simulation time.

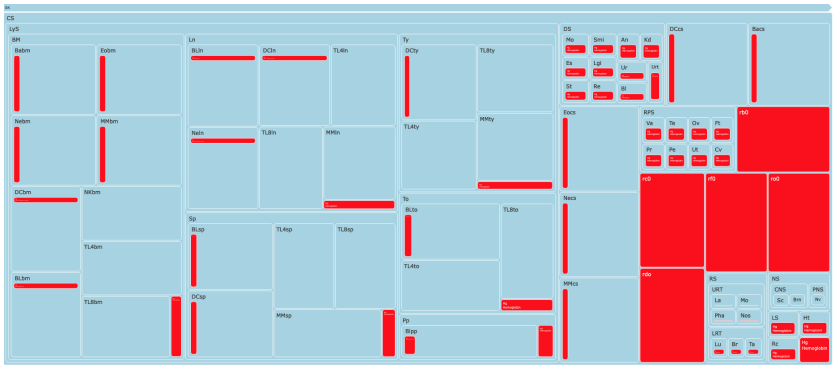

Fig. 3. A diagram obtained by the visualization tool for the membranes structure during a computation.

5 Conclusions

This work introduces significant enhancements to P systems, particularly addressing the challenges of simulating empty membranes and refining rule properties to mirror complex biological interactions more closely. The introduction of the $[\Phi]$ symbol as a representation of empty membranes aligns with biological realities where non-viable cells, lacking functional components, naturally deteriorate. This conceptual advancement not only enriches the P system's biological fidelity but also opens new avenues for simulating cellular processes with greater accuracy. The dissolution rule for empty membranes, represented by $[\Phi]_X \rightarrow \partial$, incorporates biological nuances into computational simulations.

Moreover, the elaboration on rule properties—encompassing probability, priority, execution time, and population-specific response percentages—substantially enhances the P system simulation capabilities. These properties introduce a layer of complexity that mirrors the stochastic and varied nature of biological responses across different individuals and cellular conditions. By incorporating

these aspects, our model offers a more nuanced approach that allows for a more granular and realistic representation of biological processes, acknowledging the inherent variability and stochastic nature of these systems.

The implementation of these rules through a Python-based simulator (that we have carried out in a separate work) shows the practical applicability of our theoretical advancements. The CLI-accessible simulator that we have developed not only facilitates the execution of complex simulations but also provides customizable parameters that allow researchers to tailor simulations to specific scenarios or research questions. The ability to output simulation results in various formats, including CSV for statistical analysis and YAML for detailed iterative reviews, enhances the accessibility and interpretability of simulation data.

Comparatively, while existing models in membrane computing offer robust frameworks for simulating biological processes, our approach introduces a level of specificity and adaptability previously unattained. A groundbreaking aspect of our work is the formalization of rules governing empty membranes. This novel concept not only aligns with biological realities—where cells devoid of functional components cease to exist—but also enhances the P systems ability to mimic cellular behaviors with unprecedented accuracy. By defining and subsequently dissolving empty membranes, we bridge a critical gap in the simulation of cellular dynamics.

Our proposal introduces timed rules to P systems, allowing for simulations that incorporate the dimension of time, such as requiring two hours for a specific transformation, thereby enhancing the model accuracy and realism in reflecting biological processes. This addition of time as a property to rules further refines the P systems simulations, ensuring they not only capture the sequence of biological events but also their temporal dynamics.

The visualization of membrane computing through P systems with network diagrams makes the complex interactions governed by our newly introduced rules clearer. This visual representation helps simplify the understanding of the rules, providing clear insights into the simulation process. Additionally, the active visualization of membranes and objects, along with the visible changes after applying the rules, gives a concrete view into how the simulated biological system changes over time.

In conclusion, our work not only introduces necessary extensions and innovations to P systems for simulating biological systems complexities but also lays down a foundational framework for future explorations in this domain, by enhancing rule definitions, formalizing the treatment of empty membranes, incorporating timed rules, and pioneering visual tools for system analysis.

Potential implications and future directions

The innovations presented in this study hold significant implications for the field of computational biology and beyond. By enhancing the P systems ability to simulate biological processes with higher fidelity, our work supports more accurate modeling of complex systems, such as immune responses to pathogens

or vaccines. This could, in turn, inform more precise predictive models for disease progression and treatment outcomes.

Future research could explore the integration of our enhanced P system model with machine learning algorithms to predict outcomes of biological processes under varying conditions. Additionally, expanding the model to incorporate more diverse biological rules and interactions could further its applicability across different domains of biology and medicine. Ultimately, the continued development of membrane computing models, grounded in biological accuracy and computational efficiency, will pave the way for breakthroughs in understanding and manipulating complex biological systems.

Acknowledgements

This work has received funding from CDTI within the framework of the Misiones CDTI 2021 program in the project "*Investigation of a new vaccine for human respiratory disease*" EXP - 00162016 / MIG-20211034.

References

1. Applications of membrane computing in systems and synthetic biology (2014) P. Frisco, M. Gheorghe and M.J. Pérez-Jiménez (eds.) Springer.
2. Computational and Modelling Power of P systems (2024). D. Besozzi. PhD Thesis. Universitá degli Studi di Milano.
3. P systems for biological dynamics (2006). L. Bianco, F. Fontana, G. Franco and V. Manca. In *Applications of Membrane Computing* pp 83–128. Springer.
4. Membrane computing: A wonderful framework for systems and computational biology (2023) J.M. Sempere. In *Proceedings of the Twenty-fourth International Conference on Membrane Computing* (CMC2023). L. Cienzalovà (editor), pp 4.
5. Computing with membranes (2000) Gh. Păun. Journal of Computer and System Sciences, Vol. 61, No.1 pp 108–143.
6. A guide to membrane computing (2002) Gh. Păun and Gr. Rozenberg. Theoretical Computer Science, Vol. 287, No.1 pp 73–100.
7. Respiratory syncytial virus: from pathogenesis to potential therapeutic strategies (2021) Z. Shang, S. Tan, and D. Ma. International Journal of Biological Sciences, Vol. 17, No. 14, pp 4073.
8. An introduction to immunology and immunopathology (2018). J. Marshall, R. Warrington, W. Watson and H.L. Kim, Harold. Allergy, Asthma & Clinical Immunology Vol.14, pp 1–10
9. The human immune response to respiratory syncytial virus infection (2017) C.D. Russell, S. Unger, M. Walton and J. Schwarze. Clinical microbiology reviews, Vol. 30, No.2, pp 481-502.
10. Balance entre citocinas pro y antiinflamatorias en estados sépticos (2005). R. de Pablo Sánchez, J. Monserrat Sanz, A. Prieto. Martín, E. Reyes Martín, M. Álvarez de Mon Soto, M Álvarez and M. Sánchez García. Medicina intensiva Vol. 29, No. 3, pp 151–158.
11. Developing bioinformatics computer skills (2001) C. Gibas and P. Jambeck. O'Reilly Media, Inc.

12. Computing with cells and atoms: an introduction to quantum, DNA and membrane computing (2000) C. Calude and Gh. Păun. CRC Press.
13. A Glimpse into natural computing (1999) C. Calude, Gh. Păun and M. Tatarâm. CDMTCS Research Reports CDMTCS-117. The University of Auckland, New Zealand.
14. Introduction to computational molecular biology (1997). J.C. Setubal and J. Meidanis. PWS Pub. Boston.
15. *The Oxford Handbook of Membrane Computing* (2010) Gh. Păun, G. Rozenberg, A. Salomaa, eds. Oxford University Press.