Journal of Basics and Applied Sciences Research (JOBASR)
ISSN (print): 3026-9091, ISSN (online): 1597-9962
Volume 4(1) January 2026
DOI: https://dx.doi.org/10.4314/jobasr.v4i1.19

An Application of Multiset to Deterministic P Systems with Nested Rule Prioritization and Membrane Preservation

Chinedu M. Peter[1*] & Vivian O. Ike[2]
[1,2]Department of Mathematics, Federal University, Dutsin-Ma, Nigeria.
[*]Corresponding Author Email: macpee3@yahoo.com

ABSTRACT
This paper begins with a mathematical analysis of multisets as a formal apparatus for computation: we give precise definitions, describe multiset algebra (union, intersection, difference, complementation and additive union), present the vector interpretation of multisets and Parikh-style mappings, and examine properties of multiset rewriting systems. Building on that analysis, we show how multisets can be used to encode integers (positive and negative) in a membrane system in a uniform way. Negative values being represented by an appropriate placement of multiplicities across membrane systems with only two membranes, and how algebraic properties are used in the correct design of rewriting rules. Using this multiset foundation, we construct P systems for integer arithmetic (addition, subtraction, multiplication and division) that preserve membrane structure (i.e., without dissolution) and operate under nested weak, strong and dependency priority relations. Worked examples illustrate how the multiset analyses are used in rule priorities and how priorities control the flow of computation to guarantee correctness and termination across all the cases presented.

Keywords:
Multisets,
P systems,
Rule prioritization,
Membrane preservation,
Negative integer,
Algebraic representation

INTRODUCTION

Membrane computing, introduced by Gheorghe Păun in 1998, is a branch of natural computing inspired by the structure and functioning of living cells. Its core model, the P system, captures distributed and parallel computation by means of membranes, objects, and evolution rules. Over the years, P systems have found applications in formal language theory, biology, optimization, and computational complexity. A central feature of P systems is that molecules are handled as objects of multisets. From a mathematical point of view, the use of multisets gives P systems their algebraic flexibility. The state of a membrane region can be interpreted not merely as a set, but as an algebraic structure in which objects occur with multiplicities. This makes multisets suited to encode numerical data and to perform arithmetic operations in a rigorous manner. Most existing works on arithmetic with P systems have been restricted to the use of positive integers, with objects representing quantities and rules realizing operations such as addition or multiplication in a straightforward way (Paun, 2000; Ciobanu & Angeles, 2006).

Alhazov et al. (2006) explores the use of P systems for encoding numbers and carrying out arithmetic operations, highlighting how membrane computing can simulate fundamental mathematical processes. It presents different strategies for representing integers within membranes and design rules that enable addition, subtraction, multiplication, and division to be performed in a distributed and parallel manner. Atanasiu (2001) extends this by proposing models that encode operands in base two and use membrane structures where objects represent bits that are controlled by evolution rules in parallel. It shows that these computations can achieve a lower complexity than conventional hardware implementations. Yang et al. (2015) further advances the field by presenting methods for the automatic design of P systems for arithmetic, reducing manual construction errors while preserving the inherent parallelism and distributed computation features.

Guo et al. (2013) expands arithmetic in P systems to fractions by encoding numerators and denominators as separate objects, allowing parallel manipulation for addition, subtraction, multiplication, and division of rational numbers. Zeng et al. (2012) demonstrate arithmetic in spiking neural P systems, encoding numbers as spike trains and using neuron-inspired spiking rules to leverage temporal dynamics alongside parallel computation. Guo and Chen (2008) and Guo and Zhang (2008) investigate arithmetic operations in general and single-membrane P systems, respectively, showing that even minimal membrane structures can perform addition, subtraction, multiplication, and division efficiently.

From a theoretical perspective, Freund and Păun (2003) formalizes deterministic P systems, specifying conditions for unique successor configurations and eliminating nondeterministic ambiguity, while Ibarra (2005) analyzes the distinction between deterministic and nondeterministic P systems and their implications for computational efficiency and predictability.

To achieve a more realistic mathematical representation of integers, it is necessary to incorporate both positive and negative values within the same framework. Negative integers arise naturally in real life reasonings, and without them, arithmetic remains incomplete. Multisets provide a natural and uniform way to represent both positive and negative values by distributing multiplicities of objects across distinct membranes, one can encode positive and negative values effectively.

In this paper, we develop a membrane-preserving P system that realizes the four basic integer operations, viz, addition, subtraction, multiplication, and division. Unlike approaches relying on dissolution or structural changes, our system employs nested strong rule prioritization and dependency-based rule application, ensuring deterministic and controlled computation. The inclusion of negative integers is handled seamlessly through multiset-dedicated membrane encoding, while priority mechanisms govern the flow of computation across different cases.

The main contribution of this work lies in successfully showing that positive and negative integers can be modelled using multiset representations in a single membrane structure, P systems can serve as a robust mathematical model for the structure of the integers under basic operations. This strengthens the algebraic foundations of membrane computing and demonstrates the expressive capacity of P systems as models of symbolic computation. Examples are provided to illustrate the functioning of the system and to highlight the role of rule priority in directing computations. Peter et al. (2025) and Peter (2025) are some articles on the application of multiset by the authors.

To develop a uniform multiset-based encoding of positive and negative integers within a membrane system, using object multiplicities and membrane placement to represent sign and magnitude without increasing the membrane degree or relying on membrane dissolution.

To construct deterministic, membrane-preserving P systems for the four basic arithmetic operations—addition, subtraction, multiplication, and division—on integers, and to present the corresponding computation algorithms through detailed configurations, tables, and illustrative diagrams that demonstrate rule application under mixed weak, strong, and dependency priorities.

To validate the correctness, determinism, and termination of the proposed arithmetic P systems, by analyzing the interaction of rule priorities and examining step-by-step illustrative examples that cover all sign combinations of integer operands.

JUSTIFICATION OF THE MODEL

Two notable applications of membrane computing, namely static sorting and circuit simulations were studied in Păun and Thierrin (2001). Number sorting has long been a central problem in computer science, and membrane computing provides an alternative way of simulating Boolean circuits. This naturally motivates the need to model P systems for the basic binary operations that underlie such computations.

A distinctive feature of P systems, which provides an edge over traditional digital computing, is their parallelism. Objects that match the left-hand side of a rule are applied to all the occurrences of such an object (subject to the imposed priority relation), and all membranes in the system operate simultaneously.

So far, research on arithmetic operations in P systems has focused mainly on the strong rule priority relation, likely due to its ability to capture aspects such as energy accounting and resource constraints. In contrast, the weak rule priority relation has not been given serious attention, despite its potential advantages. Păun (2000) concluded thus:

"Of course also, the weak interpretation of the priority is of interest: a rule is always used when objects exist which were not used by a rule of a higher priority."

This observation provides a strong motivation for exploring weak rule priority as well. In fact, the weak interpretation achieves greater maximality of parallelism compared to the strong interpretation. The reason is that in a weak priority system, rules of lower priority do not have to wait for a subsequent iteration if there are objects on which they can act. On the other hand, in a strong priority system, both the rules of lower priority and the objects they would act upon may be forced to wait until the next step.

Also of interest in this paper is the newly introduced dependency rule priority. It enforces that a rule is only applied if the rules it depends on have already been applied during a current iteration or transition. This is introduced in order to avoid premature rule applications which could distort the computation and lead to invalid results. Moreover, since the system is designed to preserve membranes, correctness cannot rely on dissolution. The dependency priority relation compensates for this restriction by guaranteeing that only valid sequences of dependent rules are applied. This makes the computation both stable and faithful to the model being constructed.

MATERIALS AND METHODS

RULE PRIORITIZATION

In membrane systems, priority relations are introduced to regulate the application of rules whenever multiple options are available. Weak and strong priority relations were employed. Weak priority allows a lower-priority rule to be applied provided no higher-priority rule is applicable in the same transition. Strong priority ensures that, whenever a higher-priority rule is applicable, all rules of lower priority are completely blocked, regardless of whether they could also be applied. Ciobanu and Păun (2013) explicitly defines strong priority and weak priority in P systems, explains their formal semantics, and gives examples contrasting them. Păun (2002) is one of the foundational sources of priority relations.

While these two methods of rule priority are sufficient in simple settings, they become limited where the applicability of one rule naturally depends on the prior application of another rule. To address this, the present paper introduces the notion of a *dependency* rule priority relation. Under dependency priority, the applicability of a rule is conditioned not only by its relative rank but also by whether another rule has already been triggered in the computation. This reflects situations where one transformation has no mathematical or computational meaning unless it is preceded by another. The inclusion of dependency priority thus significantly strengthens the control of the system.

This refinement also goes beyond the method used in Peter and Singh (2017) and is justified by the need to capture the extended algebraic structure considered in the present work. In this work, the weak rule relation is denoted by $>_w$, the strong rule relation is denoted by $>_s$ while the dependency rule relation is denoted by $>_d$.

SOME APPLICATIONS OF P SYSTEMS TO ARITHMETIC OPERATIONS ON INTEGERS

In this section, we design a variant of P systems for arithmetic operations on integers that incorporates dependency rule priority. Unlike the earlier model based solely on weak and strong rule priorities, the present approach enforces an dependency relation among rules called the dependency priority. This ensures that certain rules are applied only after those they depend on have been executed, thereby guaranteeing determinism throughout the computation.

The study of arithmetic operations within the framework of P systems has been a continuing line of research. For example, Yang et al. (2015) proposed arithmetic P systems based on arithmetic formula tables, where the four fundamental operations—addition, subtraction, multiplication, and division are realized by distributing digits into hierarchical membrane structures. Their approach demonstrates that P systems can serve as algorithmic devices for numerical computation, although with models that emphasize digit-wise encoding rather than multiset-based representations.

Yang et al. (2015) introduced Arithmetic P Systems based on arithmetic formula tables, where each decimal digit is placed in a separate membrane and rules mirror standard human calculation methods such as carrying and borrowing. Their approach reduces membrane complexity and provides a clear, table-driven way of handling addition, subtraction, multiplication, and division, though it is restricted to positive integers. This work is relevant to extensions like the present study, which expands arithmetic P systems to include negative integers, as their digit-wise encoding and formula-based rules can serve as a foundation for incorporating sign-handling mechanisms and broader integer arithmetic.

Other models have explored the role of priorities in achieving arithmetic tasks. Recent work has applied priority-based mechanisms to P systems that handle rational computations, such as fraction simplification, showing how deterministic results can be ensured through controlled rule application (e.g., rule priorities that enforce reductions before other transformations are executed) (see Nan et al. (2023)). Similarly, arithmetic P systems constructed on the basis of the symmetric ternary system have highlighted how careful design of rule priorities can support consistent execution of arithmetic operations (Nan et al. (2024)). These contributions highlights the importance of prioritization in aligning rule application.

Building upon these existing approaches, the present work introduces dependency rule priority as a refinement of the traditional weak and strong priority mechanisms. While weak priority allows lower-priority rules to fire if no higher-priority rule is applicable, and strong priority entirely blocks lower-priority rules in the presence of a higher-priority alternative, both remain limited in scenarios where the execution of one rule must explicitly precede the activation of another. Dependency rule priority resolves this by embedding precedence directly into the computational model, thereby strengthening determinism and broadening the expressive power of P systems in arithmetic contexts.

RESULTS AND DISCUSSION

EXTENDING P SYSTEMS TO POSITIVE AND NEGATIVE INTEGER ARITHMETIC WITH DETERMINISTIC MIXED-PRIORITY RULES

In the four models of P systems that folow only two mebranes are used, even when handling both positive and negative integers. The dependency priority relation further allows the system to avoid membrane dissolution, ensuring membrane preservation. In all the cases, m and n are the multiplicities of some objects a and b in the membrane structure. Therefore, they are positive integers. The same applies to other multiplicities used in the P system such as those of c and x. The goal is to design a P system that will carry out the four cardinal binary operations on m and n.

ADDITION P SYSTEM

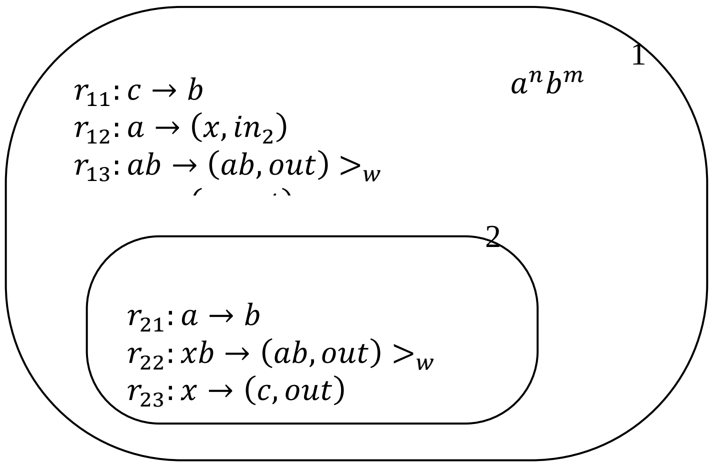
1
$a^n b^m$
$r_{11}: c \to b$
$r_{12}: a \to (x, in_2)$
$r_{13}: ab \to (ab, out) >_w$
2
$r_{21}: a \to b$
$r_{22}: xb \to (ab, out) >_w$
$r_{23}: x \to (c, out)$

Figure 1: Addition P system membrane system

Addition P system is of the form:

$$\Pi^+ = (V, \mu, (w_1, w_2), (R_1, \rho_1), (R_2, \rho_2), i_0)$$

where:

- $V = \{a, b, c, x\}$ is a finite set of objects.
- $\mu = [_1, [_2,]_2,]_1$ is a membrane structure of degree 2, with Membrane 1 (outer membrane) and, Membrane 2 (inner membrane).
- $w_1 = a^n b^m$ or $w_2 = a^n b^m$ or $w_1 = a^n$, $w_2 = b^m$ is the initial multiset of objects in Membrane 1 and/or Membrane 2, depending on the case presented.
- $R_1 = \{r_{11}: c \to b,\ r_{12}: a \to (x, in_2),\ r_{13}: ab \to (ab, out),\ r_{14}: x \to (x, out)\}$ is the set of evolution rules assigned to membranes 1.
- $R_2 = \{r_{21}: a \to b,\ r_{22}: xb \to (ab, out),$ $r_{23}: x \to (c, out)$ is the set of evolution rules assigned to Membranes 2.
- $\rho_1 = \{r_{13} >_w r_{14}\}$ is the priority relation of the rules in Membrane 1.
- $\rho_2 = \{r_{22} >_w r_{23}\}$ is the priority relation of the rules in Membrane 2.

- i_0 is the label for the output membrane. Its value is 1 if the result is positive (stored in Membrane 1), and 2 if the result is negative (stored in Membrane 2).

The idea is to compute the sum of two integers m and n, where m and n are the multiplicities of the objects in the membranes representing molecules in a biological cell.. Here, a represents the first operand, b represents the second operand while c and x are auxiliary object introduced during the computation process for communication between membranes. There are four cases considered.

Case 1: $+m$ and $+n$ Thus, $(+m)$ + $(+n)$ is positive

In this case the objects a and b with their respective multiplicities n and m are placed in Membrane 1. The rule r_{12} will be applied, it consumes the n copies of object a, produces n copies of object x and send them to Membrane 2. No other rule can be applied in Membrane 1 since only object the b is available in Membrane 1. However, there are n copies of x in Membrane 2. The rule r_{22} has a weak priority over r_{23} in Membrane 2. However, since no object b is present in Membrane 2, it will not be applied. Rather, r_{23}will be applied. It consumes the n copies of x, produces n copies of c and sends them to out of Membrane 2 to Membrane 1. It is now time for r_{12} to be applied. It converts the n copies of object c in Membrane 1 to n copies of b. We now have $n + m$ copies of b in Membrane 1. This is the result of the computation. At this stage, no other rules can be applied. The result of the computation is the $n + m$ copies of b in Membrane 1. Being in Membrane 1 means the result of the addition is positive.

Case 2: $-m$ and $+n$ Thus, $(-m)$ + $(-n)$ is negative

In this case, the two operands m and n are negative. Thus the objects and their multiplicities are placed in Membrane 2. Therefore, no rule in Membrane 1 can be applied. Only one rule in Membrane 2 will be applied and it is r_{21}. It consumes the n copies of a and produces n copies of b. The result of the computation is $n + m$ copies of b in Membrane 2.

Case 3: $-m$ and $+n$ where $n > m$. Thus, $(-m)$ + $(+n)$ is positive

In this case n copies of a ae placed in Membrane 1 while m copies of b are placed in Membrane 2. The r_{12} will be be applied in Membrane 1. It consumes the n copies of a in, produces n copies of b and sends them to Membrane 2. The rule r_{22} in Membrane 2 will now be applied. It consumes identical copies of x and b (m copies in this case), converts them to objects ab and sends them out of Membrane 2 to Membrane 1. There are now $n - m$ copies of x in Membrane 2. This is later converted to $n - m$ copies of c in and sent out of Membrane 2 by r_{23}. Meanwhile, the m copies of a and b in Membrane 1 are sent out of the membrane to the environemt by r_{13}. The $n - m$ copies of c are converted to $n - m$ copies of b in Membrane 1 where a positive result is obtained. The computation haults at this moment since no other rules can be applied.

Case 4: $-m$ and $+n$ where $n < m$. Thus, $(-m)$ + $(+n)$ is negative

This is a similar case to Case 3 above. However, the result of applying rule r_{22} is $m - n$ copies of b in Membrane 2. Therefore, no other rule will be applied in Membrane 2. And that is the result of the computation, and the result is negative since n is greater than m.

SUBTRACTION P SYSTEM

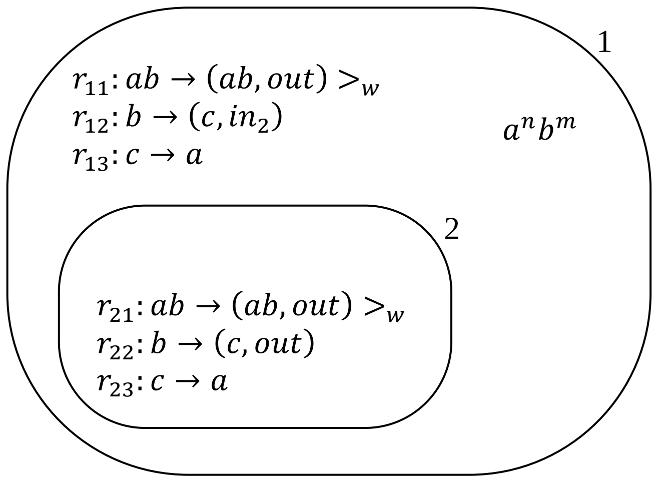
1
$r_{11}: ab \rightarrow (ab, out) >_w$
$r_{12}: b \rightarrow (c, in_2)$
$r_{13}: c \rightarrow a$
$a^n b^m$
2
$r_{21}: ab \rightarrow (ab, out) >_w$
$r_{22}: b \rightarrow (c, out)$
$r_{23}: c \rightarrow a$

Figure 2: Subtraction P system membrane system

Subtraction P system is of the form:

$$\Pi^- = (V, \mu, (w_1, w_2), (R_1, \rho_1), (R_2, \rho_2), i_0)$$

where:

- $V = \{a, b, c\}$ is a finite set of objects.
- $\mu = [_1, [_2,]_2,]_1$ is a membrane structure of degree 2, with Membrane 1 (outer membrane) and, Membrane 2 (inner membrane).
- $w = a^n b^m$, $w = a^n$ or $w = b^m$ is the initial multiset of objects in Membrane 1 or Membrane 2, depending on the case presented.
- $R_1 = \{r_{11}: ab \rightarrow (ab, out),\ r_{12}: b \rightarrow (c, in_2),$ $r_{13}: c \rightarrow a\}$ is the set of evolution rules assigned to Membrane 1.
- $R_2 = \{r_{21}: ab \rightarrow (ab, out),\ r_{22}: b \rightarrow (c, out),$ $r_{23}: c \rightarrow a$ is the set of the evolution rules assigned to Membrane 2.
- $\rho_1 = \{r_{11} >_w r_{12}\}$ is the priority relation of the rules in Membrane 1.
- $\rho_2 = \{r_{21} >_w r_{22}\}$ is the priority relation of the rules in Membrane 2.
- i_0 is the label for the output membrane. Its value is 1 if the result is positive (stored in Membrane 1), and 2 if the result is negative (stored in Membrane 2).

Case 1a: $+m$ and $+n$ where $n > m$. Thus, $(+n) - (+m)$ is positive

Since we are dealing with positive integers, the objects a and b with their respective multiplicities n and m are placed in Membrane 1. The rule r_{11} is applied. It sends m copies of a and b to the environment, leaving $n - m$ copies of a in Membrane 1. This is the result of the conputation and it encodes positive integer, since no rule can be applied again in this case.

Case 1a: $+m$ and $+n$ where $n < m$. Thus, $(+n) - (+m)$ is negative

Again Membrane 1 houses objects a and b with their multiplicities n and m, respectively. This time after the same rule r_{11} is applied there would be $m - n$ copies of the object b in Membrane 1. In this the rule r_{12} can be applied to the object b It send the $m - n$ copies of b to Membrane 2 after converting them to $c's$. Thus, we have $m - n$ copies of c in Membrane 2. No rule can be applied again. The computation halts and we have a negative result.

Case 2a: $-m$ and $-n$ where $n > m$. Thus, $(-n) - (-m)$ is negative

Since we are dealing with negative integers, the objects a and b with their respective multiplicities n and m are placed in Membrane 2. The rule r_{21} is applied. It sends n copies of a and b out of Membrane 2, leaving $n - m$ copies of b behind. The rule r_{21} will now be applied. It sends the $n - m$ copies of a out of Membrane 2. We are now left with $n - m$ copies of a in Membrane 2. This is the result of the conputation and it encodes negative integer.

Case 2b: $-m$ and $-n$ where $n < m$. Thus, $(-n) - (-m)$ is positive

Again, since we are dealing with negative integers, the objects a and b with their respective multiplicities n and m are placed in Membrane 2. The rule r_{21} is applied. It sends n copies of a and b out of Membrane 2, leaving $m-n$ copies of b behind. The rule r_{21} will now be applied. It sends the $m-n$ copies of b out of Membrane 2 after converting them to c. We now have $m-n$ copies of c in Membrane 1. This is the result of the conputation and it encodes positive integer.

Case 3a: $-m$ and n where $n > m$. Thus, $(+n) - (-m)$ is positive

In this case, n copies of a is placed in Membrane 1 while m copies of b is placed in Membrane 2. No rule can be applied in Membrane 1. The rule r_{21}, also cannot be applied in Membrane 2. Thus, the rule to be applied is r_{22}. It consumes the m copies of b, produces m copies of c and send them out of Membrane 2. There are now n copies of a and m copies of c in Membarane 1. The result of the computation is the $m+n$ copies of a in Membrane 1.

Case 3b: $-m$ and n where $n < m$. Thus, $(+n) - (-m)$ is positive

The comutation is exactly the same with that of Case 3a above.

Case 4a: m and $-n$ where $n > m$. Thus, $(-n) - (+m)$ is negative

In this case, m copies of b is placed in Membrane 1 while n copies of a is placed in Membrane 2. The rule r_{12} is applied in Membrane 1. It consumes the m copies of b, produces m copies of c and sends them to Membane 2. In Membarane 2, r_{23} will be applied. It consumes m copies of c, and produces m copies of a. There are now $m+n$ copies of a in Membrane 2. The computation halts, and the result is a negative value.

Case 4a: m and $-n$ where $n > m$. Thus, $(-n) - (+m)$ is negative

The comutation is exactly the same with that of Case 4a above.

MULTIPLICATION P SYSTEM

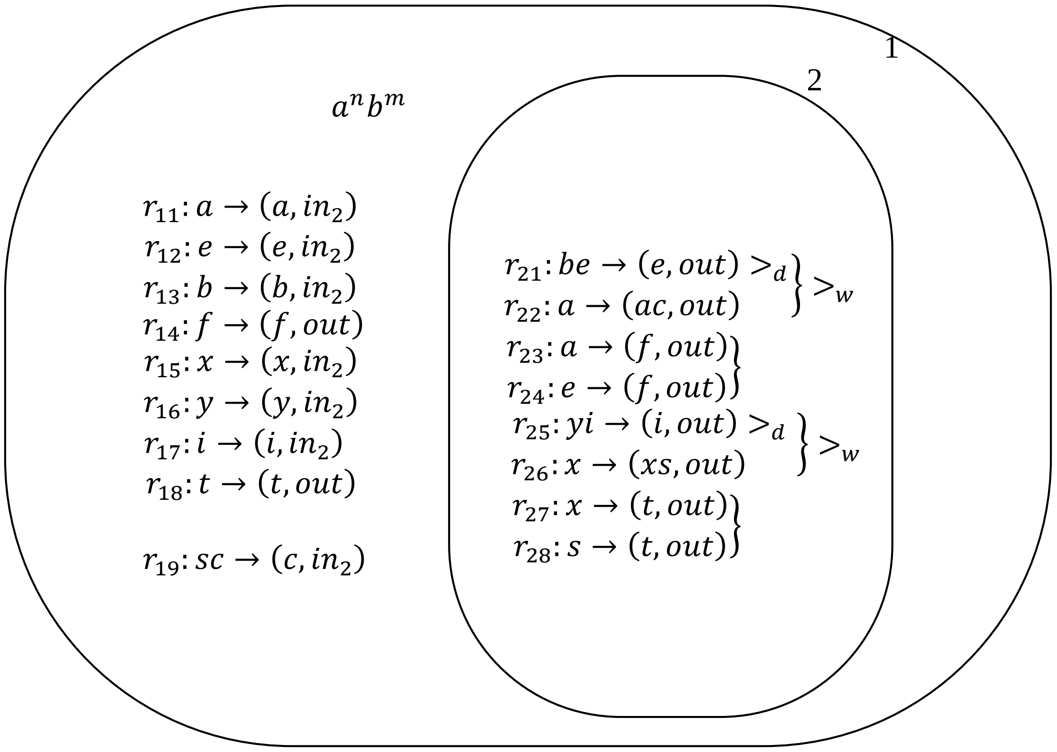
1
2
$a^n b^m$
$r_{11}: a \to (a, in_2)$
$r_{12}: e \to (e, in_2)$
$r_{13}: b \to (b, in_2)$
$r_{14}: f \to (f, out)$
$r_{15}: x \to (x, in_2)$
$r_{16}: y \to (y, in_2)$
$r_{17}: i \to (i, in_2)$
$r_{18}: t \to (t, out)$
$r_{19}: sc \to (c, in_2)$
$r_{21}: be \to (e, out) >_d$
$r_{22}: a \to (ac, out)$
$>_w$
$r_{23}: a \to (f, out)$
$r_{24}: e \to (f, out)$
$r_{25}: yi \to (i, out) >_d$
$r_{26}: x \to (xs, out)$
$>_w$
$r_{27}: x \to (t, out)$
$r_{28}: s \to (t, out)$

Figure 3: Multiplication P system membrane system

Multiplication P system is of the form:

$$\Pi^{\times} = (V, \mu, (w_1, w_2), (R_1, \rho_1), (R_2, \rho_2), i_0)$$

where:

- $V = \{a, b, c, e, f, i, s, t, x, y\}$ is a finite set of objects.
- $\mu = [_1, [_2,]_2,]_1$ is a membrane structure of degree 2, with Membrane 1 (outer membrane) and, Membrane 2 (inner membrane).
- $w = a^n b^m e$,$w = a^n$, $w = b^m$ or $w = a^n b^m e x^n y^m i$ is the initial multiset of objects in Membrane 1 or Membrane 2, depending on the case presented.
- $R_1 = \{r_{11}: a \rightarrow (a, in_2),\ r_{12}: e \rightarrow (e, in_2),$ $r_{13}: b \rightarrow (b, in_2),$ $r_{14}: f \rightarrow (f, out), r_{15}: x \rightarrow (x, in_2), r_{16}: y \rightarrow (y, in_2), r_{17}: i \rightarrow (i, in_2),$ $r_{18}: t \rightarrow (t, in_2), r_{19}: sc \rightarrow (sc, in_2)\}$ is the set of evolution rules assigned to Membrane 1.
- $R_2 = \{r_{21}: be \rightarrow (e, out),\ r_{22}: a \rightarrow (ac, out),$ $r_{23}: a \rightarrow (f, out),$ $r_{24}: e \rightarrow (f, out), r_{25}: yi \rightarrow (i, out), r_{26}: x \rightarrow (xs, out), r_{27}: x \rightarrow (t, out),$ $r_{28}: s \rightarrow (t, out)\}$ is the set of evolution rules assigned to Membrane 2.
- $\rho_1 = \emptyset$ is shows that no priority relation of rules exists in Membrane 1.
- $\rho_2 = \{r_{21} >_d r_{22}, r_{21} >_w r_{23}, r_{21} >_w r_{24}, r_{22} >_w r_{23}, r_{22} >_w r_{24}\}$ is the priority relation of the rules in Membrane 2.
- i_0 is the label for the output membrane. Its value is 1 if the result is positive (stored in Membrane 1), and 2 if the result is negative (stored in Membrane 2).

Cases 1: $(+n) \times (+m)$ is positive for positive m and n.

In this process, at the initial configuration of the P system, there are n copies of the object a and m copies of the abject b in Membrane 1. That is both a and b present in the same membrane. There is also one copy of an object e (which is always there in all cases) in Membrane 2. Since there are no existing priorities in Membrane 1. Rules that have objects can be applied. Thus, r_{11} and r_{13} will be applied in parallel on a and b in Membrane 1. They send all the multiplicities of a and/or b to Membrane 2. There are now n copies of a, m copies of b and the object e in Membrane 2. In Membrane 2, both $r_{21}$and r_{22} have a weak priority relation over r_{23} and r_{24} where r_{21} has a dependency rule priority over r_{22}. Thus r_{22} can only be applied when r_{21} is being applied and not otherwise. The rule r_{21} sends a copy of b together with a copy of e out of Membrane 2. There are now n copies of a and c and 1 copy of e in Membrane 1 and $m - 1$ copies of b in Membrane 2. In Membrane 1, the rules r_{15} and r_{16} send the n copies of a and c and the 1 copy of e back to Membrane 2. This process is repeated untill there are no copies of b left in Membrane 2. Also, the c would have appered in Membrane 1 mn times. Therefore, rules r_{21} and r_{22} will cease to be applicable. Since they have a weak priority over rules r_{23} and r_{24}, the latter two rules will be applied. They send one copy of each of a and e out of Membrane 2 after converting them to f to Membrane 1. The rule r_{14} in turn sends the copy of f out of the environment. The result of the computtion is the mn number of occurrences of the object c in Membrane 1.

Cases 2: $(-n) \times (-m)$ is positive for positive m and n.

In this case, at the initial configuration of the P system, there are n copies of the object a and m copies of the abject b in Membrane 2. That is both a and b are again present in the same membrane. Also, there is one copy of an object e (which is always there in all cases) in Membrane 2. The computation starts in Membrane 2 and continues as in Case 1 above. The result is positive at the end of the computation.

Cases 3: $(-n) \times (m)$ is negative for positive m and n.

This is a process where a and b are in different membranes. Since they are in different membranes, they are accompanied by auxiliary objects in their respective membranes by the same multiplicities of their principal objects. a is accompanied by x while b is accompanied by y. Moreover, e is accompanied by i in Membrane 2. The computation is carried out accordingly as in Process 1 above in parallel and at the same time that Process 1 is executed. The auxiliary part computation ensures that the mn copies of c and s are together converted to mn copies of c and sent back to Membrane 2 by r_{19}, ensuring that the result of the computation is negative.

DIVISION P SYSTEM

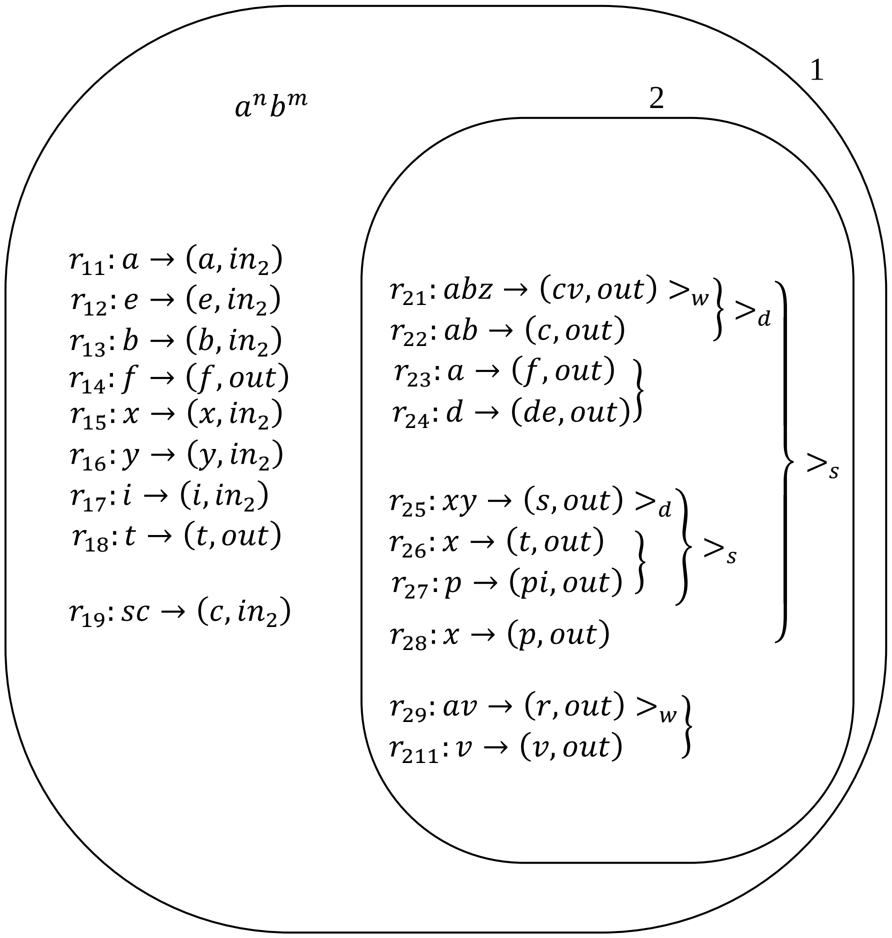
1
2
$a^n b^m$
$r_{11}: a \rightarrow (a, in_2)$
$r_{12}: e \rightarrow (e, in_2)$
$r_{13}: b \rightarrow (b, in_2)$
$r_{14}: f \rightarrow (f, out)$
$r_{15}: x \rightarrow (x, in_2)$
$r_{16}: y \rightarrow (y, in_2)$
$r_{17}: i \rightarrow (i, in_2)$
$r_{18}: t \rightarrow (t, out)$
$r_{19}: sc \rightarrow (c, in_2)$
$r_{21}: abz \rightarrow (cv, out) >_w$
$r_{22}: ab \rightarrow (c, out)$
$>_d$
$r_{23}: a \rightarrow (f, out)$
$r_{24}: d \rightarrow (de, out)$
$>_s$
$r_{25}: xy \rightarrow (s, out) >_d$
$r_{26}: x \rightarrow (t, out)$
$r_{27}: p \rightarrow (pi, out)$
$>_s$
$r_{28}: x \rightarrow (p, out)$
$r_{29}: av \rightarrow (r, out) >_w$
$r_{211}: v \rightarrow (v, out)$

Figure 4: Division P system membrane structure

A Division P system is of the form:

$$\Pi^{\div} = (V, \mu, (w_1, w_2), (R_1, \rho_1), (R_2, \rho_2), i_0)$$

where:

- $V = \{a, b, c, d, e, f, i, p, r, s, t, v, x, y, z\}$ is a finite set of objects.
- $\mu = [_1, [_2,]_2,]_1$ is a membrane structure of degree 2, with Membrane 1 (outer membrane) and, Membrane 2 (inner membrane).
- $\{w_1 = a^n b^m d, w_2 = \emptyset\}$, $\{w_1 = a^n x^n, w_2 = b^m y^m dp\}$, $\{w_1 = b^m y^m z^m, w_2 = a^n x^n dp\}$ or $\{w_1 = a^m b^m z^m, w_2 = d\}$ is the initial multiset of objects in Membrane 1 or Membrane 2, depending on the case presented.
- $R_1 = \{r_1: a \rightarrow (a, in_2),\ r_2: b \rightarrow (b, in_2),$ $r_3: x \rightarrow (x, in_2), r_4: y \rightarrow (y, in_2),$ $r_5: z \rightarrow (z, in_2), r_6: def \rightarrow (def, out), r_7: f \rightarrow (f, out), r_8: cv \rightarrow (av, in_2),$ $r_9: c \rightarrow (a, in_2), r_{10}: d \rightarrow (d, in_2), r_{11}: pit \rightarrow (pit, out), r_{12}: t \rightarrow (t, out),$ $r_{13}: s \rightarrow (x, in_2), r_{14}: p \rightarrow (p, in_2), r_{15}: ei \rightarrow (e, in_2), r_{16}: p \rightarrow (p, out),$ $r_{17}: v \rightarrow (v, out)\}$
  is the set of evolution rules assigned to Membrane 1.
- $R_2 = \{r_{21}: abz \rightarrow (cv, out),\ r_{22}: ab \rightarrow (c, out),\ r_{23}: a \rightarrow (f, out),$ $r_{24}: d \rightarrow (de, out),\ r_{25}: xy \rightarrow (s, out),$ $r_{26}: x \rightarrow (t, out),\ r_{27}: p \rightarrow (pi, out),\ r_{28}: x \rightarrow (p, out),\ r_{29}: av \rightarrow (r, out),\ r_{210}: v \rightarrow (v, out)\}$
  is the set of evolution rules assigned to Membrane 2.
- $\rho_1 = \{r_6 >_w r_7,\ r_8 >_w r_9,\ r_{11} >_w r_{12},$ $(r_6, r_7, r_8, r_9, r_{10}, r_{11}, r_{12}, r_{13}, r_{14}) >_s (r_{15}, r_{16}, r_{17})\}$ shows that no priority relation of rules exists in Membrane 1.

- $\rho_2 = \{r_{21} >_w r_{22},$
  $r_{25} >_w (r_{26}, r_{27}), (r_{21}, r_{22}) >_d (r_{23}, r_{24}),$
  $(r_{21}, r_{22}) >_d (r_{23}, r_{24}), (r_{25}, r_{26}, r_{27}) >_s r_{28},$
  $(r_{21}, r_{22}, r_{23}, r_{24}, r_{25}, r_{26}, r_{27}) >_s (r_{29}, r_{210})\}$
  is the priority relation of the rules in Membrane 2.
- i_0 is the label for the output membrane. Its value is 1 if the result is positive (stored in Membrane 1), and 2 if the result is negative (stored in Membrane 2).

Case 1: $-n$ divide by $-m$, yields a positive quotient with a negative remainder.

At the initial configuration of the P system, there are n copies of a, m copies of b and the object d in Membrane 2. The only rules that can be applied at the moment are r_{22} and r_{24}. They convert n copies of ab to c and the one copy of d to de and send them out of Membrane 2 to Membrane 1. There are now n copies of c, one copy of d and one copy of e in Membrane 1. The rules r_9 converts the object c to a and send it to Membrane 2 while the rule r_{10} sends d to Membrane 2. The process continues this way until there are no copies of b left for and r_{22} to act upon in Membrane 2. There are now copies of a left in Membrane 2. This also is converted to f and sent out of Membrane 2 by and r_{23}. There are now copies of d, e and f in Membrane 1. Therefore, and r_6 is used for the first time. It sends identical copies of d, e and f into the environment while and r_7 sends the remaining copies of f into the environment. The object c is sent back to Membrane 2 by and r_9 after converting it to a. There are now only copies of e in Membrane 1 and copies of a in Membrane 2. The multiplicity of the copies of e encodes the dividend while that of a encodes the remainder. The computation halts as no other rule can be applied.

Case 2: $-n$ divide by m, yields a negative quotient with a negative remainder.

In this case the numerator is negative while the denominator is positive. Therefore, the object a is placed in Membrane 1 while the object b is placed in Membrane 2. Moreover, since they are in separate membranes, they are accompanied by objects x in Membrane 1 and y in Membrane 2. The objects in the same membrane have the same multiplicity with the main objects. The object d as usual is placed in Membrane 2 together with its accompanying object p.The computation begins by transferring all objects in Membrane 1 to Membrane 2. The computation continues as in Case 1 above, only that this time the accompanying objects x, y and p undergo similar process as a, b and d using the rules dedicated to them. As the computation comes towards end, the availability of i in Membrane 2 makes possible for the rule r_{15} to be applied. It converts ei to e and send it to Membrane 2. Note that e in this case represents the dividend in the division. Since e would now be in Membrane 2, it shows that the result of the division is negative being in Membrane 2, so also is the remainder a.

Case 3: n divide by $-m$, yields a negative quotient with a positive remainder.

In this case the object b encoding the numerator is in Membrane 1 while the divisor is object is in Membrane 2 y and x in their respective multiplicities. An additional object z with the same multiplicity as b and y is placed in Membrane 1 as well. This is because the object encoding the dividend (b in this case) is in Membrane 1. In other words, whenever dividend object is in Membrane 1, it is accompanied by another object (z) in the same Membrane 1 with the same multiplicity. The computation takes place as usual after all objects in the initial configuration have been transferred to Membrane 2 by the required rules. This is to ensure that the remainder is transferred to Membrane 1, while the quotient is transferred to Membrane 2 as in Case 2.

Case 4: n divide by m, yields a positive dividend with a positive remainder.

At the initial configuration of the P system, objects a and b are in Membrane 1 while only the object d is in Membrane 2 as usual. Since b is in Membrane 2, it is accompanied by the object z as in Case 3 above. The computation takes place as usual. The availability of z ensures that the remainder encoded by the object r is finally in Membrane 1, while that which encode the quotient is in Membrane 1 as well.

CONCLUSION

The deterministic P systems with weak, strong and dependency rule priorities presented in this paper have successfully extended arithmetic computations from positive integers to both positive and negative integers. Several directions for future research naturally emerge from this extension.

First, it is promising to extend the present work to rational numbers. Since a rational number can be expressed as the quotient of two integers, its encoding would involve the operation of division P systems, with additional rules to preserve the sign of the result. Extending further to real numbers, rational and irrational components must be clearly distinguished. While rational numbers may be handled using integer-based division systems, encoding irrational numbers will require new strategies for both approximation and representation.

Second, fractional arithmetic could be enhanced by incorporating remainder-handling mechanisms that operate in parallel while tracking signs. This would allow P systems to deal with recurring decimals and extend applicability to scientific computation tasks.

Third, the interaction of multiple membranes can be further explored to support hierarchical arithmetic operations, such as exponentiation, roots, and modular arithmetic. Such extensions could provide powerful tools for applications in cryptography, coding theory, and symbolic computation.

Finally, the biological interpretation of objects can be deepened. For example, distinguishing between computational objects (representing integer values) and cofactor objects (regulating computation flow) may strengthen the analogy between membrane systems and biochemical processes. This could lead to models that are simultaneously computationally rigorous and biologically possible.

In conclusion, the results presented here provide a strong foundation for extending deterministic P systems toward a full arithmetic operation and expression encompassing integers, rationales, and reals, while also inviting cross-disciplinary applications in mathematics, computer science, and systems biology.

ACKNOWLEDGEMENTS

We extend our deep appreciation to Gh. Păun, the pioneer of membrane computing, whose foundational contributions continue to inspire this line of research. We are also grateful to the authors of the various texts and resources we have consulted in the preparation of this work, as well as to colleagues and individuals who have contributed directly or indirectly to the development of the ideas presented in this paper. Our special thanks go to the anonymous reviewers whose constructive comments and suggestions have greatly improved the clarity, structure, and overall quality of this manuscript.

REFERENCE

Alhazov, A., Freund, R., Oswald, M., & Verlan, S. (2007, September). Partial halting in P systems using membrane rules with permitting contexts. In *International Conference on Machines, Computations, and Universality* (pp. 110-121). Berlin, Heidelberg: Springer Berlin Heidelberg.

Atanasiu, A. (2001). *Arithmetic with membranes*. Romanian Journal of Information Science and Technology, 4(1–2), 5–20.

Ciobanu, G., & Angeles, C. (2006). Encodings and arithmetic operations in membrane computing. In Theory and Applications of Models of Computation (TAMC 2006) (pp. 621–630). Springer.

Ciobanu, G., & Păun, G. (2013). "Using the Formal Framework for P Systems" in *Membrane Computing: 14th International Conference, CMC 2013, Revised Selected Papers*. Springer.

Freund, R., & Păun, G. (2003). *On deterministic P systems* (Technical report). Retrieved from http://psystems.disco.unimib.it

Guo, P., & Chen, J. (2008, May). Arithmetic Operation in Membrane System. In *Proceedings of the 2008 International Conference on Biomedical Engineering and Informatics* (Vol. 1, pp. 231-234). IEEE. https://doi.org/10.1109/BMEI.2008.13

Guo, P., & Zhang, H. (2008, December). Arithmetic operation in single membrane. In *Proceedings of the 2008 International Conference on Computer Science and Software Engineering* (CSSE) (Vol. 3, pp. 532-535). IEEE. https://doi.org/10.1109/CSSE.2008.121

Guo, P., Zhang, H., Chen, H., & Chen, J. (2013). Fraction arithmetic operations performed by P systems. *Chinese Journal of Electronics, 22*(4), 690–694. https://doi.org/10.1049/cje.2013.07.012

Ibarra, O. H. (2005). Some computational issues in membrane computing. In *Mathematical Foundations of Computer Science 2005* (MFCS 2005) (Lecture Notes in Computer Science, Vol. 3618, pp. 39-51). Springer. https://doi.org/10.1007/11549345_4

Nan, H., Xue, Z., Li, C., Zhou, M., & Liu, X. (2023). P system design for integer factorization. *Applied Sciences, 13*(15), 8910. https://doi.org/10.3390/app13158910

Nan, H., Zhang, J., Guo, P., Jiang, J., & Zhang, X. (2024). An arithmetic operation P system based on symmetric ternary system. *PLOS ONE, 19*(11), e0312778. https://doi.org/10.1371/journal.pone.0312778

Păun, G. (2000). Computing with membranes (P systems). *Journal of Computer and System Sciences, 61(1), 108–143.* Academia.

Paun, G. (2002). *Membrane computing: an introduction*. Springer Science & Business Media..

Păun, G., & Thierrin, G. (2001). Multiset Processing by Means of Systems of Finite State Transducers. In O. Boldt & H. Jürgensen (Eds.), Automata Implementation. *WIA 1999 (Lecture Notes in Computer Science, Vol. 2214, pp. 140-157). Springer. https://doi.org/10.1007/3-540-45526-4_13*

Peter, C. M., Singh, D. (2017) Arithmetic operations in deterministic P systems based on the weak rule priority, *Proceedings of the Nigerian Computer Society*, Vol. 28, No. 10, 92-100.

Peter, C., (2025) On Singh's Dressed Epsilon Perspective of Multigroup. *Journal of Basics and Applied Sciences Research (JOBASR)* Vol 3(3), 1597-9962.

Peter, C., Balogun, F., Adewumi A. O. (2025) On Substructures and Root Sets in Antimultigroups and their Direct Products., *Journal of Basics and Applied Sciences Research (JOBASR)* Vol 3(4), 1597-9962.

Yang, R., Guo, P., & Li, J. (2015). Arithmetic P Systems Based on Arithmetic Formula Tables. *Chinese Journal of Electronics*, *24*(3), 542-549. DOI:10.1049/cje.2015.07.018.

Zeng, X., Song, T., Zhang, X., & Pan, L. (2012). Performing four basic arithmetic operations with spiking neural P systems. *IEEE Transactions on NanoBioscience*, *11*(4), 366-374. https://doi.org/10.1109/TNB.2012.2211034