



**CDMTCS**  
**Research**  
**Report**  
**Series**



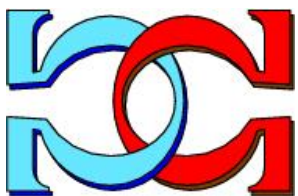
**Structured Modelling with**  
**Hyperdag P Systems: Part A**



**Radu Nicolescu**  
**Michael J. Dinneen**  
**Yun-Bum Kim**

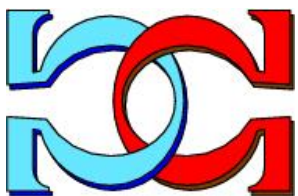


Department of Computer Science,  
University of Auckland,  
Auckland, New Zealand



CDMTCS-342  
December 2008

(minor revision May 2009)



Centre for Discrete Mathematics and  
Theoretical Computer Science

# Structured Modelling with Hyperdag P Systems: Part A

Radu Nicolescu, Michael J. Dinneen and Yun-Bum Kim  
Department of Computer Science, University of Auckland,  
Private Bag 92019, Auckland, New Zealand

May 2009

## Abstract

P systems provide a computational model based on the structure and interaction of living cells [9]. A P system consists of a hierarchical nesting of cell-like membranes, which can be visualised as a rooted tree.

Although the P systems are computationally complete, many real world models, e.g., from socio-economic systems, databases, operating systems, distributed systems, seem to require more expressive power than provided by tree structures. Many such systems have a primary tree-like structure completed with shared or secondary communication channels. Modelling these as tree-based systems, while theoretically possible, is not very appealing, because it typically needs artificial extensions that introduce additional complexities, inexistent in the originals.

In this paper we propose and define a new model that combines structure and flexibility, called *hyperdag P systems*, in short, *hP systems*, which extend the definition of conventional P systems, by allowing dags, interpreted as hypergraphs, instead of trees, as models for the membrane structure.

We investigate the relation between our hP systems and neural P systems. Despite using an apparently restricted structure, i.e., a dag instead of a general graph, we argue that hP systems have essentially the same computational power as tissue and neural P systems. We argue that hP systems offer a structured approach to membrane-based modelling that is often closer to the behavior and underlying structure of the modelled objects.

Additionally, we enable dynamical changes of the rewriting modes (e.g., to alternate between determinism and parallelism) and of the transfer modes (e.g., the switch between unicast or broadcast). In contrast, classical P systems, both tree and graph based P systems, seem to focus on a statical approach.

We support our view with a simple but realistic example, inspired from computer networking, modelled as an hP system with a shared communication line (broadcast channel). In Part B of this paper we will explore this model further and support it with a more extensive set of examples.

**Keywords:** P systems, hyperdag P systems, tissue P systems, neural P systems, membrane structures.

# 1 Introduction

P systems provide a distributed computational model, based on the structure and interaction of living cells, introduced by G. Păun in 1998 [8]. The model was initially based on transition rules, but was later expanded into a large family of related models. Essentially, all versions of P systems have a structure consisting of cell-like membranes and a set of rules that govern their evolution over time.

Many of the “classical” versions use a structure where membranes correspond to nodes in a rooted tree. Such a structure is often visualised as Venn diagram where nesting denotes a parent/child relationship. For example, Figure 1 [10] shows the same P system structure with 9 membranes, labeled as  $1, \dots, 9$ , both as a rooted tree and as a Venn diagram.

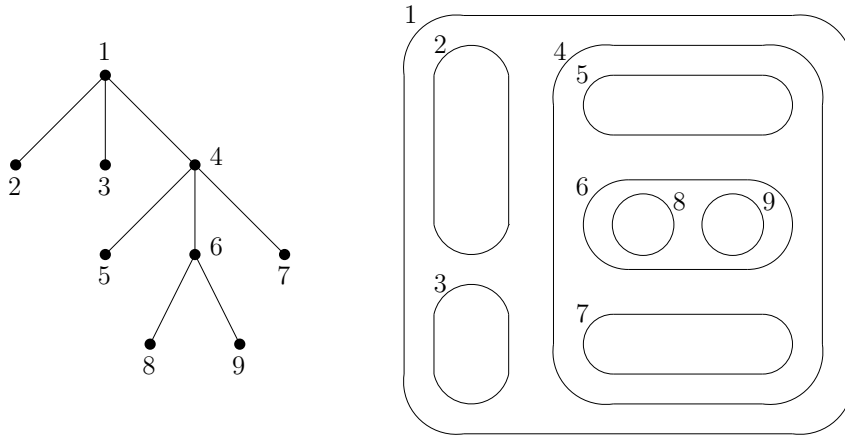


Figure 1: A P system structure represented as a tree and as a Venn diagram.

More, recently, tissue P systems [7] and neural P systems [9], here abbreviated as tP and nP systems, respectively, have been introduced, partially to overcome the limitations of the tree model. Essentially, these systems organize their cells in an arbitrary digraph. For example, ignoring for the moment the actual contents of cells (states, objects, rules), Figure 2 illustrates the membrane structure of a simple tP or nP system, consisting of 3 cells,  $\sigma_1, \sigma_2, \sigma_3$ , where cell  $\sigma_1$  is designated as the output cell.

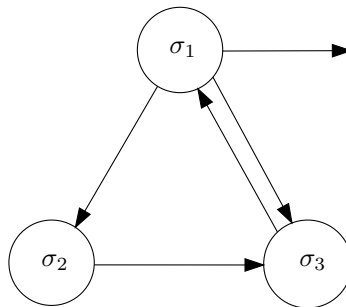


Figure 2: A tP/nP system structure represented as a digraph.

A large variety of rules have been used to describe the operational behavior of P systems, the main ones being: multiset rewriting rules, communication rules and membrane handling rules. Essentially, transition P systems and nP systems use multiset rewriting rules, P systems with symport/antiport operate by communicating immutable objects, P systems with active membranes combine all three type rules. For a comprehensive overview and more details, we refer the reader to [9, 10].

Besides theoretical computer science and biology, P systems have been applied to a variety of other domains, ranging from linguistics [5] to theoretically efficient solutions of NP-complete problems [14], or to model distributed algorithms [3, 6]. The underlying tree structure provides good support for reasoning and formal verification, good potential for efficient implementation on multi-core architectures, and an excellent visualisation, very appealing to practitioners.

Although the P systems are computationally complete, many real world models seem to require more expressive power, essentially trees augmented by shared or secondary communication channels. For example, the notion of a processing node having an unique parent is not true for (a) computer networks where a computer could simultaneously be attached to several subnets (e.g., to an Ethernet bus and to a wireless cell), (b) living organisms may be the result of multiple inheritance (e.g., the evolutionary “tree” is not really a tree, because of lateral gene transfer [4]) and (c) socio-economic scenarios where a player is often connected to and influenced by more than one factors [11, 12, 13].

Modelling these as tree-based systems, while theoretically possible, is not very appealing. Simulating shared or secondary channels requires artificial mechanisms that will ripple data up and down the tree, via a common ancestor. This could of course limit the merits of using a formal model. Tissue and neural P systems have been introduced to model such cases [7, 9]; details on neural P systems, in short, *nP systems*, are given in Section 3. However, these extensions are based on general graphs and, while allowing any direct communications, they also tend to obscure the structures already present in the modelled objects, limiting the advantages that a more structured approach could provide. Verification is more difficult without a clear modularisation of concerns, practical parallel implementation could be less efficient, if the locality of reference is not enforced, and visualisations are not very meaningful, unless the primary structure is clearly emphasized.

We do not think that we have to choose between structure and flexibility. We propose a solution that seems to combine both, i.e., flexibility without sacrificing the advantages of a structured approach.

Our main contribution in this paper is to propose a new model for P systems, called *hyperdag P systems*, in short, *hP systems*, that allows more flexible communications than tree-based models, while preserving a strong hierarchical structure. This model, defined in Section 4, (a) extends the tree structure of classical P systems to directed acyclic graphs (dags), (b) augments the operational rules of nP systems with broadcast facilities (via a *go-sibling* transfer tag), and (c) enables dynamical changes of the rewriting modes (e.g., to alternate between determinism and parallelism) and of the transfer modes (e.g., to switch between unicast or broadcast). In contrast, classical P systems, both tree and graph based P systems, seem to focus on a statical approach.

We investigate the relation between our hP systems and neural P systems. Despite

using an apparently less powerful structure, we show in Section 5 that our simple dag model has the same computational power as graph-based tissue and neural P systems.

We argue that hP systems offer a structured approach to membrane-based modelling that is often closer to the behavior and underlying structure of the modelled objects. Because our extensions address the membrane topology, not the rules model, they can be applied to a variety of P system flavors, including transition systems and symport/antiport systems.

We support our view with a realistic example (see Examples 23 and 24), inspired from computer networking, modelled as an hP system with a shared communication line (broadcast channel).

Classical P systems allow a “nice” planar visualisation, where the parent/child relationships between membranes are represented by Venn-like diagrams. We show in Section 6 that the extended membrane structure of hP systems can still be visualised by hierarchically nested planar regions.

In this article we will restrict ourselves to P systems based on multiset rewriting rules, such as used by transition P systems and nP systems. However, because our extensions address the membrane topology, not the rules model, they can be applied to a variety of other P system flavors.

## 2 Preliminaries

A (binary) *relation*  $R$  over two sets  $X$  and  $Y$  is a subset of their Cartesian product,  $R \subseteq X \times Y$ . For  $A \subseteq X$  and  $B \subseteq Y$ , we set  $R(A) = \{y \in Y \mid \exists x \in A, (x, y) \in R\}$ ,  $R^{-1}(B) = \{x \in X \mid \exists y \in B, (x, y) \in R\}$ .

A *digraph* (directed graph)  $G$  is a pair  $(X, A)$ , where  $X$  is a finite set of elements called *nodes* (or *vertices*), and  $A$  is a binary relation  $A \subseteq X \times X$ , of elements called *arcs*. For an arc  $(x, y) \in A$ ,  $x$  is a *predecessor* of  $y$  and  $y$  is a *successor* of  $x$ . A length  $n - 1$  *path* is a sequence of  $n$  distinct nodes  $x_1, \dots, x_n$ , such that  $\{(x_1, x_2), \dots, (x_{n-1}, x_n)\} \subseteq A$ . A *cycle* is a path  $x_1, \dots, x_n$ , where  $n \geq 1$  and  $(x_n, x_1) \in A$ .

A *dag* (directed acyclic graph) is a digraph  $(X, A)$  without cycles. For  $x \in X$ ,  $A^{-1}(x) = A^{-1}(\{x\})$  are  $x$ 's *parents*,  $A(x) = A(\{x\})$  are  $x$ 's *children*,  $A(A^{-1}(x)) \setminus \{x\} = A(A^{-1}(\{x\})) \setminus \{x\}$  are  $x$ 's *siblings* (siblings defines a symmetric relation). A node  $x \in X$  is a *source* iff  $|A^{-1}(x)| = 0$ , and  $x \in X$  is a *sink* iff  $|A(x)| = 0$ . The *height* of a node  $x$  is the maximum length of all paths from  $x$  to a sink node. An arc  $(x, y)$  is *transitive* if there exists a path  $x_1, \dots, x_n$ , with  $x_1 = x$ ,  $x_n = y$  and  $n > 2$ . Dags without transitive arcs are here called *canonical*.

A (rooted unordered) *tree* is a dag with exactly one source, called *root*, and all other nodes have exactly one parent (predecessor). Sinks in a tree are also called *leaves*. A *topological order* of a dag is a linear reordering of vertices, in which each vertex  $x$  comes before all its children vertices  $A(x)$ .

Dags and trees are typically represented with parent-child arcs on the top-down axis, i.e., sources/roots up and sinks/leaves down. Figure 3 shows a simple dag.

We consider a variant hypergraph definition, based on multisets, as an extension of

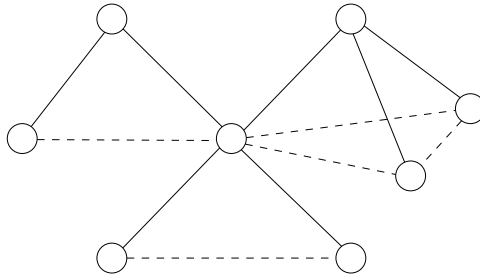


Figure 3: A simple dag. The parent-child axis is up-down. Here, plain lines indicate parent-child relations and dashed lines indicate siblings.

the classical definition [1], which is based on sets. A *hypergraph* is here a pair  $(X, E)$ , where  $X$  is a finite set of elements called *nodes* (or *vertices*), and  $E$  is a finite *multiset* of subsets of  $X$ , i.e.,  $e \in E \Leftrightarrow e \subseteq X$ . By using a multiset of edges, instead of a more conventional set of edges, we introduce an *intensional* element, where two *extensionally* equivalent hyperedges (i.e., hyperedges containing the same nodes) are not necessarily equal. A *graph* is a set based hypergraph where hyperedges are known as *edges* and contain exactly two nodes. Alternatively, a graph  $(X, E)$  can be interpreted as a digraph  $(X, A)$ , where  $A = \{(x, y) \mid \{x, y\} \in E\}$ . Hypergraphs (set or multiset based) can be represented by planar diagrams, where hyperedges are represented as regions delimited by images of Jordan curves (simple closed curves) [2].

With the above hypergraph definition, a height 1 dag  $(X, A)$  can be interpreted as a hypergraph  $(X, E)$ , where  $E$  is the multiset  $E = \{A(x) \mid |A^{-1}(x)| = 0\}$ . For example, Figure 4 represents, side by side, the dag  $D = (\{a, b, c, d, e, f\}, \{(d, a), (d, b), (d, c), (e, b), (e, c), (f, b), (f, c)\})$  and its corresponding hypergraph  $H = (\{a, b, c\}, \{d, e, f\})$ , where  $d = \{a, b, c\}, e = \{b, c\}, f = \{b, c\}$ . Note that the apparently empty differences of regions are needed in the case of *multiset* based hypergraphs, to support the *intensional* (as opposed to the *extensional*) aspect: here  $e \neq f$ , despite containing the same nodes,  $b$  and  $c$ , and neither  $e$  nor  $f$  is included in  $d$ .

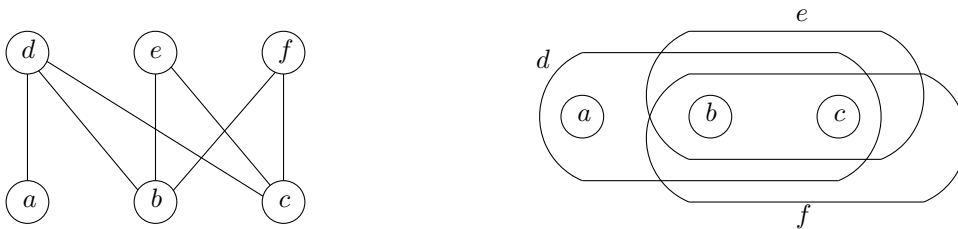


Figure 4: A simple height 1 dag and its corresponding hypergraph representation.

Generalizing the above hypergraph definition, a height  $n$  *generalized hypergraph* is a system  $(X, E)$ , recursively built via a sequence of  $n$  hypergraphs  $(X_1, E_1), \dots, (X_n, E_n)$  where  $X_1 = X$ ,  $X_i \cap E_i = \emptyset$ ,  $X_{i+1} = X_i \cup E_i$ ,  $e \cap E_i \neq \emptyset$  for  $\forall e \in E_{i+1}$  and  $E = \bigcup_{i \in \{1, \dots, n\}} E_i$ . An arbitrary height  $n$  dag can be represented by a height  $n$  generalized hypergraph, where the hypergraph nodes correspond to dag sinks, and height  $i$  hyperedges

correspond to height  $i$  dag nodes, for  $i \in \{1, \dots, n\}$ .

We will later see that any generalized hypergraph that corresponds to a non-transitive dag can also be represented by hierarchically nested planar regions delimited by Jordan curves, where arcs are represented by direct nesting. For example, Figure 5 shows a height 2 dag and its corresponding height 2 hypergraph  $(X, E)$ , where  $X = X_1 = \{a, b, c, d, e\}$ ,  $E_1 = \{f, g, h\}$ ,  $E_2 = \{i\}$ ,  $E = \{f, g, h, i\}$ .

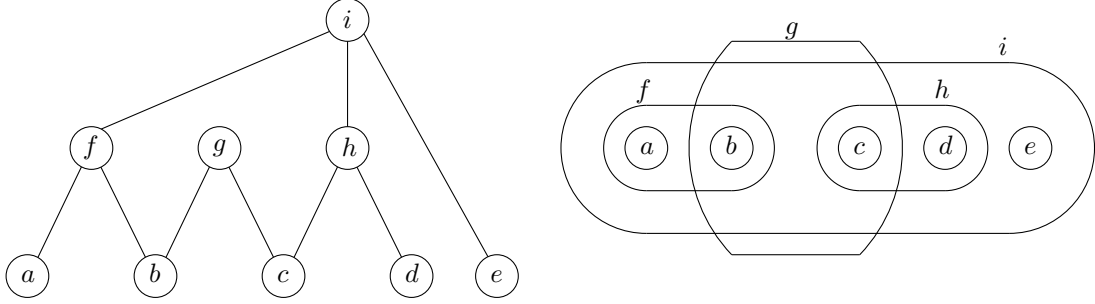


Figure 5: A height 2 dag and its corresponding height 2 hypergraph.

An *alphabet*  $O$  is a finite non-empty sets of *objects*. We will assume that the alphabet  $O$  is implicitly ordered. *Multisets* over an alphabet  $O$  are represented as strings over  $O$ , such as  $o_1^{n_1} \dots o_k^{n_k}$ , where  $o_i \in O$ ,  $n_i \geq 0$ , and, in the canonical form, letters appear in sorted order, i.e.,  $o_1 < \dots < o_k$ , and  $n_i \geq 1$ . The set of all multisets is denoted by  $O^*$ . For this representation, two strings are equivalent if they become equal after sorting, e.g.,  $a^2cbd^0a$  and  $a^3bc$  are equivalent representations of the same multiset  $\{a, a, a, b, c\}$ . Under this convention, the empty string  $\lambda$  represents the empty multiset, and string concatenation represents multiset union, e.g.,  $(a^2c) \cdot (ab) = a^3bc$ .

### 3 Neural P Systems

In this paper we present the definition of neural P systems as given in [9], that coincides with an early definition of tissue P systems as given in [7]. We define the following sets of tagged objects:  $O_{go} = \{(a, go) \mid a \in O\}$ ,  $O_{out} = \{(a, out) \mid a \in O\}$ , and we set  $O_{tot} = O \cup O_{go} \cup O_{out}$ . For simplicity, we will use subscripts for these tagged objects, such as  $a_{go}$  for  $(a, go)$  and  $a_{out}$  for  $(a, out)$ . We also define projection homomorphisms, here denoted in postfix notation:  $|_O, |_{go}, |_{out} : O_{tot}^* \rightarrow O^*$ , by  $o|_O = o, o_{go}|_{go} = o, o_{out}|_{out} = o$  for  $o \in O$ , and otherwise  $\lambda$ . For example,  $a^2a_{go}^3b^4b_{go}|_{go} = a^3b$ .

**Definition 1 (Neural P systems [9, 7])** A *neural P system* (of degree  $m \geq 1$ ) is a system  $\Pi = (O, \sigma_1, \dots, \sigma_m, syn, i_{out})$ , where:

1.  $O$  is an ordered finite non-empty alphabet of *objects*;
2.  $\sigma_1, \dots, \sigma_m$  are *cells*, of the form  $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i)$ ,  $1 \leq i \leq m$ , where:
  - $Q_i$  is a finite set (of *states*),

- $s_{i,0} \in Q_i$  is the *initial state*,
  - $w_{i,0} \in O^*$  is the *initial multiset* of objects,
  - $P_i$  is a finite set of multiset rewriting *rules* of the form  $sx \rightarrow s'x'y_{go}z_{out}$ , where  $s, s' \in Q_i$ ,  $x, x' \in O^*$ ,  $y_{go} \in O_{go}^*$  and  $z_{out} \in O_{out}^*$ , with the restriction that  $z_{out} = \lambda$  for all  $i \in \{1, \dots, m\} \setminus \{i_{out}\}$ .
3.  $syn$  is a set of *digraph* arcs on  $\{1, \dots, m\}$ , i.e.,  $syn \subseteq \{1, \dots, m\} \times \{1, \dots, m\}$ , representing *unidirectional* communication channels between cells, known as *synapses*;
  4.  $i_{out} \in \{1, \dots, m\}$  indicates the *output cell*, the only cell allowed to send objects to the “environment”.

**Example 2** To illustrate the operational behavior of nP systems, consider again the example of Figure 2, expanded now with states, rules and objects, see Figure 6. For simplicity, in this example only cell  $\sigma_1$  provides rules. More formally, this nP system can be defined as the system  $\Pi_1 = (O, \sigma_1, \sigma_2, \sigma_3, syn, i_{out})$ , where:

- $O = \{a, b, c, d\}$ ;
- $\sigma_1 = (\{s, t\}, s, a^2, \{sa \rightarrow sdb_{go}c_{go}, sa \rightarrow sd, s \rightarrow t, td \rightarrow td_{out}\})$ ;
- $\sigma_2 = (\{s\}, s, \lambda, \emptyset)$ ;
- $\sigma_3 = (\{s\}, s, \lambda, \emptyset)$ ;
- $syn = \{(1, 2), (1, 3), (2, 3), (3, 1)\}$ ;
- $i_{out} = 1$ .

Neural P systems operate as indicated by multiset rewriting rules. A rewriting rule takes the existing state and objects and generates a new state and new objects, where some of the generated objects are tagged for *communication*, i.e., for transfer to neighboring cells along existing synapses. Objects that need to be transferred to a neighboring cell are tagged with *go* and objects that need to be output in the environment are tagged with *out* (in this definition, this is only possible from the  $i_{out}$  node).

**Definition 3 (Configurations [9, 7])** A *configuration* of the nP system  $\Pi$  is an  $m$ -tuple of the form  $(s_1w_1, \dots, s_mw_m)$ , with  $s_i \in Q_i$  and  $w_i \in O^*$ , for  $1 \leq i \leq m$ . The  $m$ -tuple  $(s_{1,0}w_{1,0}, \dots, s_{m,0}w_{m,0})$  is the *initial configuration* of  $\Pi$ .

**Example 4** For example, the initial configuration of the nP system  $\Pi_1$  in Figure 6, is  $C_0 = (saa, s\lambda, s\lambda)$ .



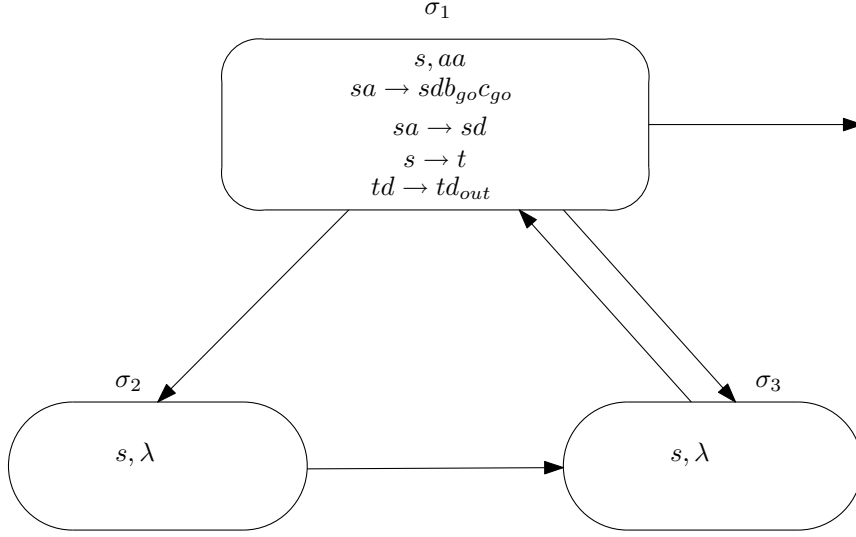


Figure 6:  $\Pi_1$ , a simple nP system with states, objects and rules.

**Definition 5 (Rewrite and transfer modes [9, 7])** Neural P systems have three *modes of rewriting objects*, inside a cell, *min* (minimum), *par* (parallel), *max* (maximum), and three *modes of transferring objects*, from a given cell to another cell, *repl* (replicate), *one*, *spread*. For each nP system, the rewriting and transfer modes are *fixed* from start and apply to all rewriting and transition steps, as defined below.

**Definition 6 (Rewriting steps [9, 7])** For each cell  $\sigma_i$  with  $s, s' \in Q_i$ ,  $x \in O^*$ ,  $y \in O_{tot}^*$ , we define a *rewriting step*, denoted by  $\Rightarrow_\alpha$ , where  $\alpha \in \{min, par, max\}$ :

- $sx \Rightarrow_{min} s'y$  iff  $sw \rightarrow s'w' \in P_i$ ,  $w \subseteq x$ , and  $y = (x - w) \cup w'$ ;
- $sx \Rightarrow_{par} s'y$  iff  $sw \rightarrow s'w' \in P_i$ ,  $w^k \subseteq x$ ,  $w^{k+1} \not\subseteq x$ , for some  $k \geq 1$ , and  $y = (x - w^k) \cup w'^k$ ;
- $sx \Rightarrow_{max} s'y$  iff  $sw_1 \rightarrow s'w'_1, \dots, sw_k \rightarrow s'w'_k \in P_i$ ,  $k \geq 1$ , such that  $w_1 \dots w_k \subseteq x$ ,  $y = (x - w_1 \dots w_k) \cup w'_1 \dots w'_k$ , and there is no  $sw \rightarrow s'w' \in P_i$ , such that  $w_1 \dots w_k w \subseteq x$  (note that rules selected arbitrarily can be combined only if they start from the same state  $s$  and end in the same state  $s'$ ).

**Example 7** As an example, considering cell 1 from the nP system  $\Pi_1$ , illustrated in Figure 6, the following rewriting steps are possible:

- $sa^n d^k \Rightarrow_{min} sa^{n-1} b_{go} c_{go} d^k$
- $sa^n d^k \Rightarrow_{min} sa^{n-1} d^{k+1}$
- $sa^n d^k \Rightarrow_{min} ta^n d^k$

- $ta^n d^k \Rightarrow_{min} ta^n d^{k-1} d_{out}$
- $sa^n d^k \Rightarrow_{par} sb_{go}^n c_{go}^n d^{k+n}$
- $sa^n d^k \Rightarrow_{par} sd^{k+n}$
- $sa^n d^k \Rightarrow_{par} ta^n d^k$
- $ta^n d^k \Rightarrow_{par} ta^n d_{out}^k$
- $sa^n d^k \Rightarrow_{max} sb_{go}^l c_{go}^l d^{k+n}$  with  $0 \leq l \leq n$
- $sa^n d^k \Rightarrow_{max} ta^n d^k$
- $ta^n d^k \Rightarrow_{max} ta^n d_{out}^k$

We now define a *transition step between two configurations*, denoted by  $\Rightarrow_{\alpha,\beta}$ , where  $\alpha$  is an *object processing mode* and  $\beta$  is an *object transfer mode*. Essentially, for a transition step we apply a rewriting step in each cell and we send to the neighbors all objects tagged for transfer.

**Definition 8 (Transition steps, adapted from [9, 7])** Given two configurations  $C_1 = (s_1 w_1, \dots, s_m w_m)$  and  $C_2 = (s'_1 w'_1, \dots, s'_m w'_m)$ , we write  $C_1 \Rightarrow_{\alpha,\beta} C_2$ , for  $\alpha \in \{min, par, max\}$ ,  $\beta \in \{repl, one, spread\}$ , if the conditions below are met.

First, we apply rewriting steps (as defined in Definition 6) on each cell, i.e.,  $s_i w_i \Rightarrow_{\alpha} s'_i w'_i$ ,  $1 \leq i \leq m$ .

Secondly, we define  $z_{j,k}$ , the outgoing object multisets from  $j$  to  $k$ , where  $j \in \{1, \dots, m\}$  and  $k \in syn(j)$ :

- If  $\beta = repl$ , then
  - $z_{j,k} = w'_j|_{go}$ , for  $k \in syn(j)$ ;
- If  $\beta = one$ , then
  - $z_{j,k_j} = w'_j|_{go}$ , for an arbitrary  $k_j \in syn(j)$ , and  $z_{j,k} = \lambda$  for  $k \in syn(j) \setminus \{k_j\}$ ;
- If  $\beta = spread$ , then
  - $\{z_{j,k}\}_{k \in syn(j)}$  is an arbitrary multiset partition of  $w'_j|_{go}$ .

Finally, we set  $w''_i = w'_i|_O \cup \bigcup_{j \in syn^{-1}(i)} z_{j,i}$ , for  $i \in \{1, \dots, m\}$ .

**Example 9** As an illustration, considering again the nP system  $\Pi_1$ , given in Figure 6, the following are examples of possible transfer steps:

- $(sa^n d^k, s, s) \Rightarrow_{min, repl} (sa^{n-1} d^{k+1}, sbc, sbc)$

- $(sa^n d^k, s, s) \Rightarrow_{min, repl} (sa^{n-1} d^{k+1}, s, s)$
- $(sa^n d^k, s, s) \Rightarrow_{min, one} (sa^{n-1} d^{k+1}, sbc, s)$
- $(sa^n d^k, s, s) \Rightarrow_{min, one} (sa^{n-1} d^{k+1}, s, sbc)$
- $(sa^n d^k, s, s) \Rightarrow_{min, spread} (sa^{n-1} d^{k+1}, sbc, s)$
- $(sa^n d^k, s, s) \Rightarrow_{min, spread} (sa^{n-1} d^{k+1}, sb, sc)$
- $(sa^n d^k, s, s) \Rightarrow_{min, spread} (sa^{n-1} d^{k+1}, sc, sb)$
- $(sa^n d^k, s, s) \Rightarrow_{min, spread} (sa^{n-1} d^{k+1}, s, sbc)$

**Definition 10 (Halting and results [9, 7])** If no more transitions are possible, the nP system halts. For halted nP system  $\Pi$ , the *computational result* is the multiset that was cumulatively sent *out* (to the “environment”) from the output cell  $i_{out}$ . The *numerical result* is the vector  $N_{\alpha, \beta}(\Pi)$  consisting of the object multiplicities in the multiset result, where  $\alpha \in \{min, par, max\}$  and  $\beta \in \{repl, one, spread\}$ . Within the family of P systems, two systems are *functionally equivalent* if they yield the same computational results.

**Example 11** For example, if a nP system  $\Pi$ , over the alphabet  $\{a, b, c, d\}$ , sends out the multiset  $a^2 cd^3$  and then halts, then its numerical result is vector  $N_{\alpha, \beta}(\Pi) = (2, 0, 1, 3)$ .

**Example 12** We replicate here another, perhaps more interesting example, originally given in [7]. Consider the following nP system, see Figure 7,  $\Pi_2 = (O, \sigma_1, \sigma_2, \sigma_3, syn, i_{out})$ , where:

- $O = \{a\}$ ;
- $\sigma_1 = (\{s\}, s, a, \{sa \rightarrow sa_{go}, sa \rightarrow sa_{out}\})$ ;
- $\sigma_2 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{go}\})$ ;
- $\sigma_3 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{go}\})$ ;
- $syn = \{(1, 2), (1, 3), (2, 1), (3, 1)\}$ ;
- $i_{out} = 1$ .

The following results are straightforward:

$$\begin{aligned}
N_{min, repl}(\Pi_2) &= \{(n) \mid n \geq 1\}, \\
N_{min, \beta}(\Pi_2) &= \{(1)\}, \text{ for } \beta \in \{one, spread\}, \\
N_{par, repl}(\Pi_2) &= \{(2^n) \mid n \geq 0\}, \\
N_{par, \beta}(\Pi_2) &= \{(1)\}, \text{ for } \beta \in \{one, spread\}, \\
N_{max, repl}(\Pi_2) &= \{(n) \mid n \geq 1\}, \\
N_{max, \beta}(\Pi_2) &= \{(1)\}, \text{ for } \beta \in \{one, spread\}.
\end{aligned}$$

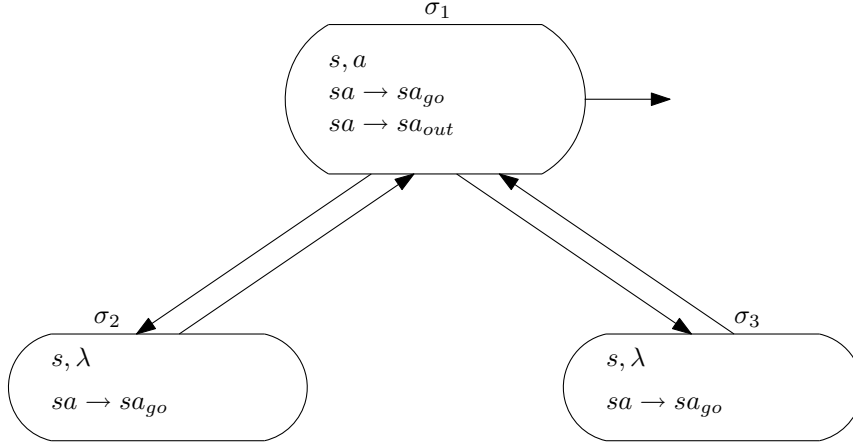


Figure 7:  $\Pi_2$ , another simple nP system.

## 4 Hyperdag P Systems

We define hP systems as essentially nP systems (see Section 3), where the underlying digraph is a dag, with several adjustments. Besides the existing *go*, *out* tags, we consider three other object tags:

1. *go-parent*, abbreviated by the symbol  $\uparrow$ , indicating objects that will be sent to parents;
2. *go-child*, abbreviated by the symbol  $\downarrow$ , indicating objects that will be sent to children;
3. *go-sibling*, abbreviated by the symbol  $\leftrightarrow$ , indicating objects that will be sent to siblings;

The precise semantics of these tags will be explained below when we detail the hP object transfer modes. In fact, we could also discard the *go* tag, as it corresponds to the union of these news tags (*go-parent*, *go-child*, *go-sibling*); however, we will keep it here, for its concise expressive power. We use similar notation as nP systems for these new tags  $O_\uparrow, O_\downarrow, O_{\leftrightarrow}$ , and postfix projections  $|\uparrow, |\downarrow, |\leftrightarrow$ .

Other extension tags, including addressing mechanisms (such as *from/to/via* tags) are possible, and indeed seem natural, but this is beyond the scope of this article.

**Definition 13 (Hyperdag P systems)** A *hP system* (of degree  $m$ ) is a system  $\Pi = (O, \sigma_1, \dots, \sigma_m, \delta, I_{out})$ , where:

1.  $O$  is an ordered finite non-empty alphabet of *objects*;
2.  $\sigma_1, \dots, \sigma_m$  are *cells*, of the form  $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i)$ ,  $1 \leq i \leq m$ , where:
  - $Q_i$  is a finite set (of *states*),

- $s_{i,0} \in Q_i$  is the *initial state*,
  - $w_{i,0} \in O^*$  is the *initial multiset* of objects,
  - $P_i$  is a finite set of multiset rewriting *rules* of the form  $sx \rightarrow s'x'u\uparrow v\downarrow w\leftrightarrow y_{go}z_{out}$ , where  $s, s' \in Q_i$ ,  $x, x' \in O^*$ ,  $u\uparrow \in O_\uparrow^*$ ,  $v\downarrow \in O_\downarrow^*$ ,  $w\leftrightarrow \in O_\leftrightarrow^*$ ,  $y_{go} \in O_{go}^*$  and  $z_{out} \in O_{out}^*$ , with the restriction that  $z_{out} = \lambda$  for all  $i \in \{1, \dots, m\} \setminus I_{out}$ ,
3.  $\delta$  is a set of dag parent/child arcs on  $\{1, \dots, m\}$ , i.e.,  $\delta \subseteq \{1, \dots, m\} \times \{1, \dots, m\}$ , representing *bidirectional* communication channels between cells;
  4.  $I_{out} \subseteq \{1, \dots, m\}$  indicates the *output cells*, the only cells allowed to send objects to the “environment”.

The essential novelty of our proposal is to replace the arbitrary arc set *syn* by a more structured arcs set  $\delta$  (dag), or, otherwise interpreted, as a generalized multiset-based hypergraph. This interpretation has actually suggested the name of our proposal, hyperdag P systems, and their abbreviation hP.

The changes in the rules format are mostly adaptations needed by the new topological structure. Here we have reused and enhanced the rewriting rules used by nP systems [9]. However, we could adopt and adapt any other rule set, from other variants or extensions of P systems, such as, rewriting, antiport/symport or boundary rules [10].

Definitions of *configurations*, *transitions*, *computations* and *results of computations* in hP systems are similar to definitions used for nP systems (Section 3), with the following essential additions/differences, here informally stated:

- The *rewrite mode*  $\alpha$  and *transfer mode*  $\beta$  could but need not be fixed from the start—they may vary, for each cell  $\sigma_i$  and state  $s \in Q_i$ .
- If *object transfer mode* is *repl* (this is a deterministic step):
  - the objects tagged with  $\uparrow$  will be sent to all the parents, replicated as necessary
  - the objects tagged with  $\downarrow$  will be sent to all the children, replicated as necessary
  - the objects tagged with  $\leftrightarrow$  will be sent to all the siblings, of all sibling groups, replicated as necessary
- If *object transfer mode* is *one* (this is a nondeterministic step):
  - the objects tagged with  $\uparrow$  will be sent to one of the parents, arbitrarily chosen
  - the objects tagged with  $\downarrow$  will be sent to one of the children, arbitrarily chosen
  - the objects tagged with  $\leftrightarrow$  will be sent to one of the siblings, of one of the sibling groups, arbitrarily chosen
- If *object transfer mode* is *spread* (this is a nondeterministic step):
  - the objects tagged with  $\uparrow$  will be split into submultisets and distributed among the parents, in an arbitrary way

- the objects tagged with  $\downarrow$  will be split into submultisets and distributed among the children, in an arbitrary way
- the objects tagged with  $\leftrightarrow$  will be split into submultisets and distributed among the siblings and sibling groups, in an arbitrary way

Figure 8 schematically shows the possible transfers of objects from a membrane  $i$ , having two children, two parents, hence two sibling groups, with one sibling in the first group and two siblings in the other. The above mentioned transfer modes will select one, some or all the illustrated transfer targets, deterministically (*repl*) or nondeterministically (*one*, *spread*).

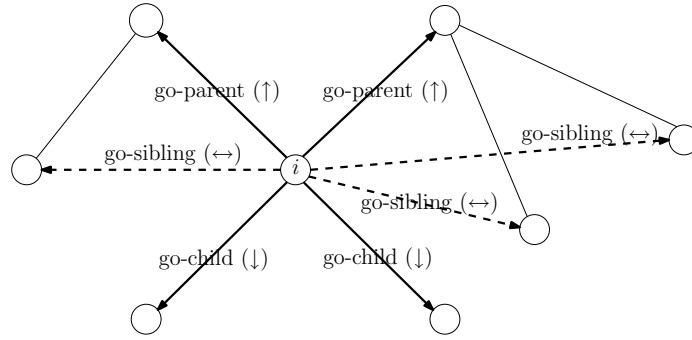


Figure 8: Transfer modes in an hP system. The parent-child axis is top-down. Plain lines indicate parent-child relations and dashed lines indicate siblings. Arrows at the end of long thick lines, plain or dashed, indicate possible transfer directions from node  $i$ .

More formal definitions follow.

**Definition 14 (Configurations)** A *configuration* of the hP system  $\Pi$  is an  $m$ -tuple of the form  $(s_1w_1, \dots, s_mw_m)$ , with  $s_i \in Q_i$  and  $w_i \in O^*$ , for  $1 \leq i \leq m$ . The  $m$ -tuple  $(s_{1,0}w_{1,0}, \dots, s_{m,0}w_{m,0})$  is the *initial configuration* of  $\Pi$ .

**Definition 15 (Rewrite and transfer modes)** For an hP system of degree  $m$ ,

- the *object rewriting mode* is a function

$$\alpha : \bigcup_{i \in \{1, \dots, m\}} \{i\} \times Q_i \rightarrow \{min, par, max\} .$$

- the *object transfer mode* is a function

$$\beta : \bigcup_{i \in \{1, \dots, m\}} \{i\} \times Q_i \rightarrow \{repl, one, spread\} .$$

**Definition 16 (Rewriting steps)** For each cell  $\sigma_i$  with  $s, s' \in Q_i$ ,  $x \in O^*$ ,  $y \in O_{tot}^*$ , we define a *rewriting step*, denoted by  $\Rightarrow_\alpha$ , where  $\alpha = \alpha(i, s) \in \{min, par, max\}$ .

- $sx \Rightarrow_{min} s'y$  iff  $sw \rightarrow s'w' \in P_i$ ,  $w \subseteq x$ , and  $y = (x - w) \cup w'$ ;
- $sx \Rightarrow_{par} s'y$  iff  $sw \rightarrow s'w' \in P_i$ ,  $w^k \subseteq x$ ,  $w^{k+1} \not\subseteq x$ , for some  $k \geq 1$ , and  $y = (x - w^k) \cup w'^k$ ;
- $sx \Rightarrow_{max} s'y$  iff  $sw_1 \rightarrow s'w'_1, \dots, sw_k \rightarrow s'w'_k \in P_i$ ,  $k \geq 1$ , such that  $w_1 \dots w_k \subseteq x$ ,  $y = (x - w_1 \dots w_k) \cup w'_1 \dots w'_k$ , and there is no  $sw \rightarrow s'w' \in P_i$ , such that  $w_1 \dots w_k w \subseteq x$  (note that rules can be combined only if they start from the same state  $s$  and end in the same state  $s'$ ).

**Definition 17 (Transition steps)** Given two configurations  $C_1 = (s_1 w_1, \dots, s_m w_m)$  and  $C_2 = (s'_1 w''_1, \dots, s'_m w''_m)$ , we write  $C_1 \Rightarrow_{\alpha, \beta} C_2$ , for  $\alpha$  and  $\beta$  (as defined in Definition 15) if the conditions below are met.

First, we apply rewriting steps (as defined in Definition 16) on each cell, i.e.,  $s_i w_i \Rightarrow_{\alpha(i, s_i)} s'_i w'_i$ ,  $1 \leq i \leq m$ .

Secondly, we define  $z_{j,k}^\uparrow$ ,  $z_{j,k}^\downarrow$ ,  $z_{j,k}^\leftrightarrow$ , the outgoing multisets from  $j$  to  $k$ , where  $j \in \{1, \dots, m\}$  and, respectively,  $k \in \delta^{-1}(j)$ ,  $k \in \delta(j)$ ,  $k \in \delta(\delta^{-1}(j)) \setminus \{j\}$ :

- If  $\beta(j, s_j) = repl$ , then
  - $z_{j,k}^\uparrow = w'_j|_\uparrow$ , for  $k \in \delta^{-1}(j)$ ;
  - $z_{j,k}^\downarrow = w'_j|_\downarrow$ , for  $k \in \delta(j)$ ;
  - $z_{j,k}^\leftrightarrow = w'_j|_{\leftrightarrow}$ , for  $k \in \delta(\delta^{-1}(j)) \setminus \{j\}$ .
- If  $\beta(j, s_j) = one$ , then
  - $z_{j,k_j}^\uparrow = w'_j|_\uparrow$ , for an arbitrary  $k_j \in \delta^{-1}(j)$ , and  $z_{j,k}^\uparrow = \lambda$  for  $k \in \delta^{-1}(j) \setminus \{k_j\}$ ;
  - $z_{j,k_j}^\downarrow = w'_j|_\downarrow$ , for an arbitrary  $k_j \in \delta(j)$ , and  $z_{j,k}^\downarrow = \lambda$  for  $k \in \delta(j) \setminus \{k_j\}$ ;
  - $z_{j,k_j}^\leftrightarrow = w'_j|_{\leftrightarrow}$ , for an arbitrary  $k_j \in \delta(\delta^{-1}(j)) \setminus \{j\}$ , and  $z_{j,k}^\leftrightarrow = \lambda$  for  $k \in \delta(\delta^{-1}(j)) \setminus \{j, k_j\}$ .
- If  $\beta(j, s_j) = spread$ , then
  - $\{z_{j,k}^\uparrow\}_{k \in \delta^{-1}(j)}$  is an arbitrary multiset partition of  $w'_j|_\uparrow$ ;
  - $\{z_{j,k}^\downarrow\}_{k \in \delta(j)}$  is an arbitrary multiset partition of  $w'_j|_\downarrow$ ;
  - $\{z_{j,k}^\leftrightarrow\}_{k \in \delta(\delta^{-1}(j)) \setminus \{j\}}$  is an arbitrary multiset partition of  $w'_j|_{\leftrightarrow}$ .

Finally, we set  $w''_i = w'_i|_O \cup \bigcup_{j \in \delta^{-1}(i)} z_{j,i}^\uparrow \cup \bigcup_{j \in \delta(i)} z_{j,i}^\downarrow \cup \bigcup_{j \in \delta(\delta^{-1}(i)) \setminus \{i\}} z_{j,i}^\leftrightarrow$ , for  $i \in \{1, \dots, m\}$ .

**Definition 18 (Halting and results)** If no more transitions are possible, the hP system halts. For halted hP system, the *computational result* is the multiset that was cumulatively sent *out* (to the “environment”) from the output cells  $I_{out}$ . The *numerical result* is the set of vectors consisting of the object multiplicities in the multiset result.

**Example 19** As examples, consider two hP systems,  $\Pi_3$  and  $\Pi_4$ , both functional equivalent a functional equivalent of the earlier  $\Pi_2$  nP system.

$\Pi_3 = (O, \sigma_1, \sigma_2, \sigma_3, \delta, I_{out})$ , see Figure 9, where:

- $O = \{a\}$ ;
- $\sigma_1 = (\{s\}, s, a, \{sa \rightarrow sa_{\downarrow}, sa \rightarrow sa_{out}\})$ ;
- $\sigma_2 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{\uparrow}\})$ ;
- $\sigma_3 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{\uparrow}\})$ ;
- $\delta = \{(1, 2), (1, 3)\}$ ;
- $I_{out} = \{1\}$ .

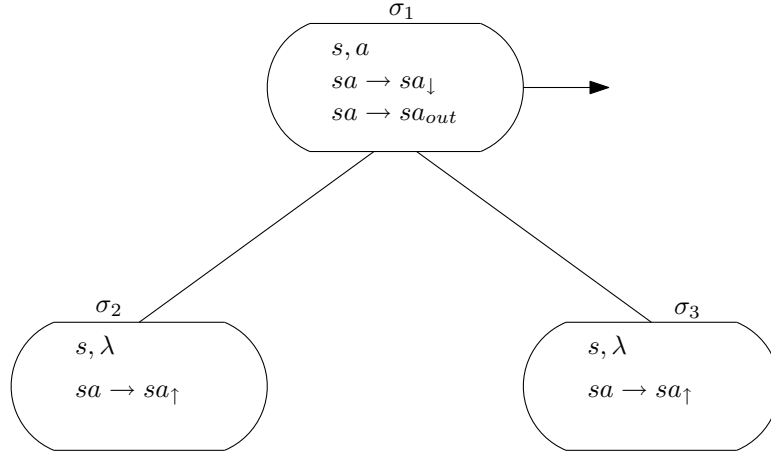


Figure 9:  $\Pi_3$ , a simple hP system (equivalent to the  $\Pi_2$  nP system of Figure 7).

$\Pi_4 = (O, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \delta, I_{out})$ , see Figure 10, where:

- $O = \{a\}$ ;
- $\sigma_1 = (\{s\}, s, a, \{sa \rightarrow sa_{\leftrightarrow}, sa \rightarrow sa_{out}\})$ ;
- $\sigma_2 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{\leftrightarrow}\})$ ;
- $\sigma_3 = (\{s\}, s, \lambda, \{sa \rightarrow sa_{\leftrightarrow}\})$ ;
- $\sigma_4 = (\{s\}, s, \lambda, \emptyset)$ ;
- $\sigma_5 = (\{s\}, s, \lambda, \emptyset)$ ;
- $\delta = \{(4, 1), (4, 2), (5, 1), (5, 3)\}$ ;
- $I_{out} = \{1\}$ .



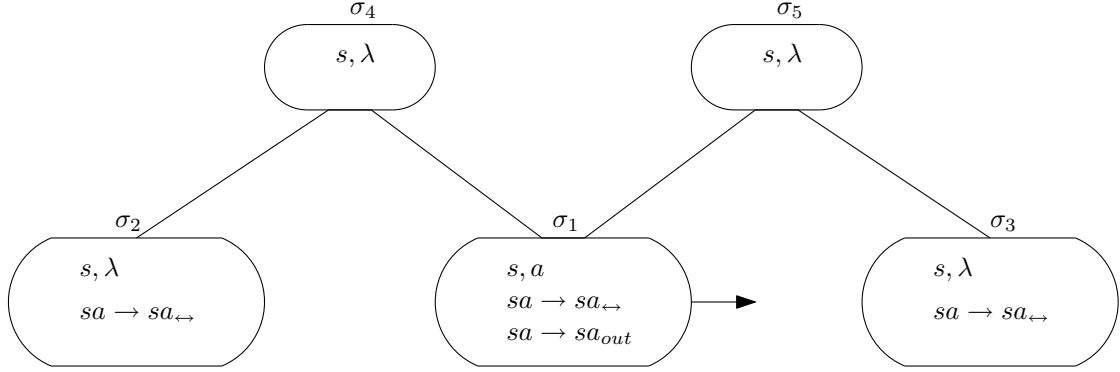


Figure 10:  $\Pi_4$ , another simple hP system (equivalent to the  $\Pi_2$  nP system of Figure 7).

## 5 Relations between P Systems, Neural P Systems and Hyperdag P Systems

### Theorem 20 (Hyperdag P systems include non-dissolving P systems)

Any non-dissolving<sup>1</sup> transition P system can be simulated by an hP system, with the same number of steps and object transfers.

**Proof:** Given a non-dissolving, transition P system  $\Pi$  [10], we build a functionally equivalent hP system  $\Pi_H$  by the following transformation  $f$ . Essentially, we use the same elements, with minor adjustments. As the underlying structure, we can reuse the rooted tree structure of the P systems, because any rooted tree is a dag.

$\Pi = (O, C, \mu, w_1, \dots, w_m, R_1, \dots, R_m, i_o)$ ,  $\Pi_H = f(\Pi) = (O', \sigma'_1, \dots, \sigma'_m, \delta, I_{out})$ .

- $O' = O$ ;
- $\sigma'_i = (Q'_i, s'_{i,0}, w'_{i,0}, P'_i)$ ,  $1 \leq i \leq m$ , where:
  - $Q'_i = \{s\}$ , where  $s$  is any symbol  $\notin O$ ;
  - $s'_{i,0} = s$ ;
  - $w'_{i,0} = w_i$ ;
  - $P'_i = \{su \rightarrow sv' \mid u \rightarrow v \in R_i\}$ , where  $v'$  is a translation of  $v$  by the following homomorphism:  $(O \cup O \times \{here, in, out\})^* \rightarrow O^*$ , such that  $a \rightarrow a$ ,  $(a, here) \rightarrow a$ ,  $(a, out) \rightarrow a_{\uparrow}$ ,  $(a, in) \rightarrow a_{\downarrow}$ .
- $\delta = \mu$ ;
- $I_{out} = \{i_o\}$ ;

<sup>1</sup>A dissolving membrane occurs if we allow rules that tell a membrane to disappear where its remaining objects go to its parent membrane; see [10].

- The object rewrite mode is the *max* constant function, i.e.,  $\alpha(i, s) = \text{max}$ , for  $i \in \{1, \dots, m\}, s \in Q_i$ ;
- The object transfer mode is the *spread* constant function, i.e.,  $\beta(i, s) = \text{spread}$ , for  $i \in \{1, \dots, m\}, s \in Q_i$ .

Tags  $\text{go-child}(\downarrow)$ ,  $\text{go-parent}(\uparrow)$  correspond to P system target indications *in*, *out*, respectively. An empty tag corresponds to P system target indication *here*. Object rewrite and transfer modes of hP systems are a superset of object rewrite and transfer mode of P systems.

We omit here the rest of the proof that shows the two systems,  $\Pi$  and  $\Pi_H$ , yield the same computational results, which is now straightforward but lengthy.  $\square$

**Remark 21** In analogy to the P systems, it is straightforward to extend the hP systems with additional features, such as dissolving membranes, priorities or polarities. However, to keep the arguments simple, we refrain of considering such extensions.

Proving that hP systems also cover nP systems appears more daunting. However, here we will use a natural interpretation of hP systems, where the bulk of the computing will be done by the sink nodes, and the upper nodes (parents) will function mostly as communication channels.

**Remark 22** The combination of *go-sibling* ( $\leftrightarrow$ ) with *repl* object transfer mode enable the efficient modelling of a communication *bus*, using only one hyperedge or, in the corresponding dag,  $n$  arcs. In contrast, any formal systems that use graph edges (or digraph arcs) to model 1:1 communication channels will need  $n(n-1)$  separate edges (or  $2n(n-1)$  arcs) to model the associated complete subgraph (clique). It is expected that this modelling improvement will also translate into a complexity advantage, if we use the number of messages measure. In hP systems, a local broadcast needs only one message to siblings, while needing  $n-1$  messages in graph/digraph based systems.

**Example 23** Figure 11 shows the structure of an hP system that models a computer network. Four computers are connected to “Ethernet Bus 1”, the other four computers are connected to “Ethernet Bus 2”, while two of the first group and two of the second group are at the same time connected to a wireless cell. In this figure we also suggest that “Ethernet Bus 1” and “Ethernet Bus 2” are themselves connected to a higher level communication hub, in a generalized hypergraph.

**Example 24** Figure 12 shows the computer network of Figure 11, modelled as a graph (if we omit arrows) or as a digraph (if we consider the arrows). Note that the graph/digraph models, such as nP, do not support the grouping concept, i.e., there is no direct way to mark the nodes  $a, b, c, d$  as being part of the “Ethernet Bus 1”, etc.

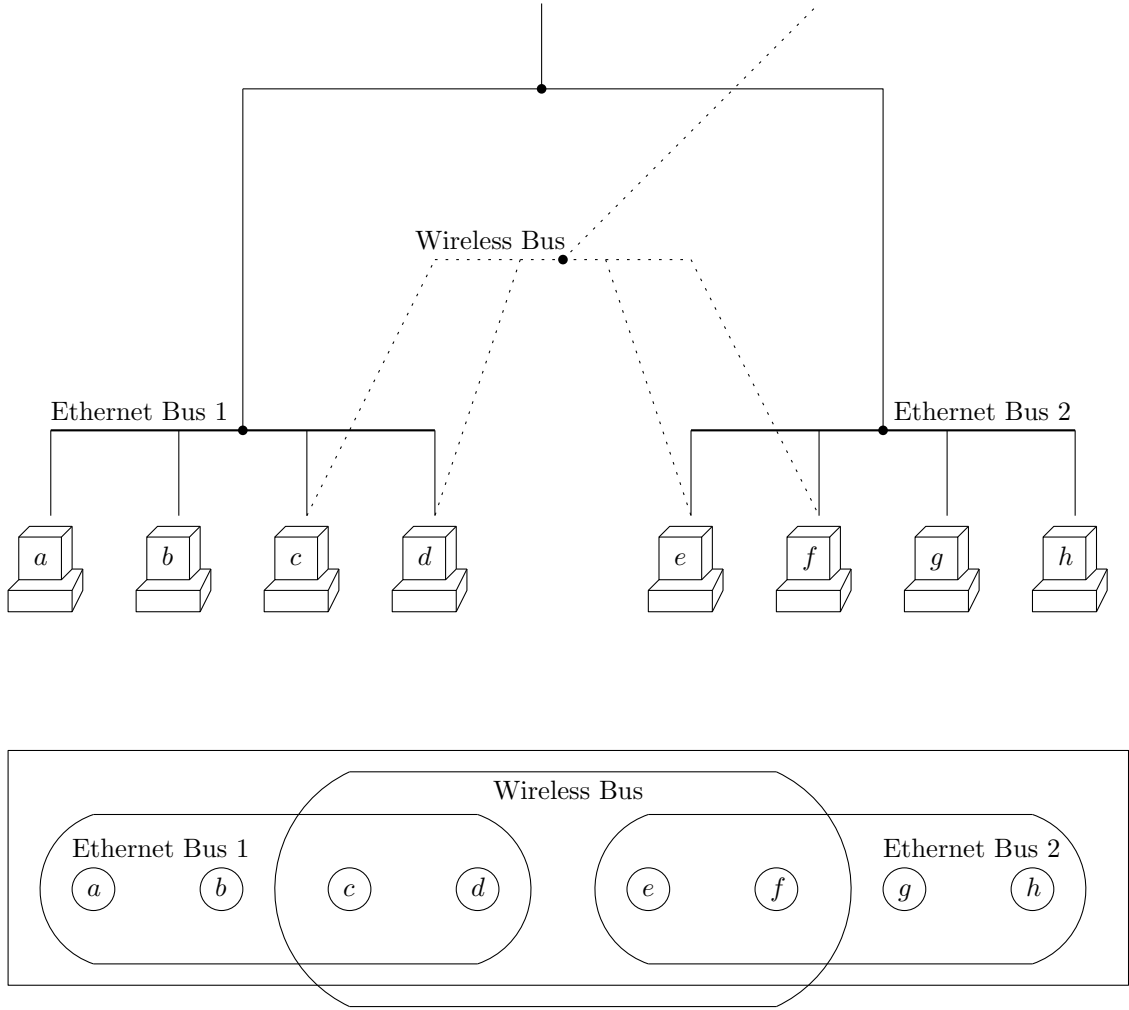


Figure 11: A computer network and its corresponding hP/hypergraph representation.

We can now sketch the proof of the theorem comparing hP systems and nP systems.

**Theorem 25 (Hyperdag P systems can simulate bidirectional nP systems)**

*Any bidirectional nP system can be simulated by an hP system, with the same number of steps and object transfers.*

**Proof:**

Given a bidirectional nP system  $\Pi_N$ , we build a functionally equivalent hP system  $\Pi_H$  by the following transformation  $f$ . As the underlying structure, we use a dag of height 1, where the cells are sink nodes, and the *syn* arcs are reified as height 1 nodes.

Without loss of generality, we assume that in the nP systems synapses are distinct from cells.

$$\Pi_N = (O, \sigma_1, \dots, \sigma_m, \text{syn}, i_{out}), \Pi_H = f(\Pi_N) = (O', \sigma'_1, \dots, \sigma'_{m+|\text{syn}|}, \delta, I_{out}).$$

- $O' = O$ ;

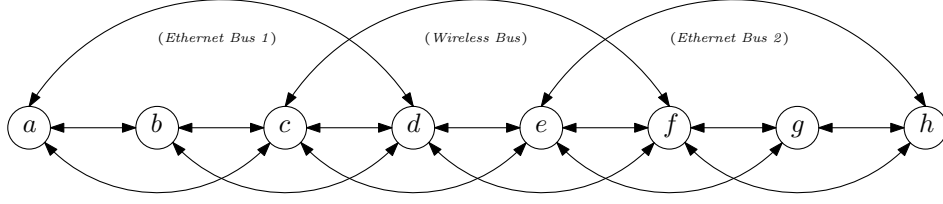


Figure 12: The graph/digraph representations of the computer network of Figure 11.

- $\sigma'_i = f(\sigma_i)$ , for  $i \in \{1, \dots, m\}$ , where:
  - $Q'_i = Q_i$ ;
  - $s'_{i,0} = s_{i,0}$ ;
  - $w'_{i,0} = w_{i,0}$ ;
  - $P'_i = \{u \rightarrow v' \mid u \rightarrow v \in P_i\}$ , where  $v'$  is a translation of  $v$  by the following homomorphism:  $O_{tot}^* \rightarrow O^*$ , such that  $a \rightarrow a$ ,  $a_{go} \rightarrow a_{\leftrightarrow}$ ,  $a_{out} \rightarrow a_{out}$  ;
- $\sigma'_{m+1}, \dots, \sigma'_{m+|syn|}$  is an arbitrary ordering of elements in  $syn$ ;
- $\delta = \{(e, u), (e, v) \mid e = (u, v) \in syn\}$ ;
- $I_{out} = \{i_{out}\}$ ;
- The object rewrite mode is a constant function, i.e.,  $\alpha(i, s) = \alpha_0$ , for  $i \in \{1, \dots, m+|syn|\}$ ,  $s \in Q_i$ , where  $\alpha_0 \in \{min, par, max\}$ ;
- The object transfer mode is a constant function, i.e.,  $\beta(i, s) = \beta_0$ , for  $i \in \{1, \dots, m+|syn|\}$ ,  $s \in Q_i$ , where  $\beta_0 \in \{repl, one, spread\}$ .

Here the nodes corresponding to synapses are inactive as they just link neighboring cells.

Essentially, the cells keep their original nP rules, with minor adjustments. A *go* tag in nP rules corresponds to a sibling( $\leftrightarrow$ ) tag in hP rules. Object rewrite and transfer modes of hP systems are a superset of object rewrite and transfer modes of nP systems.

We omit here the rest of the proof that shows the two systems,  $\Pi_N$  and  $\Pi_H$ , yield the same computational results, which is now straightforward but lengthy. □

**Remark 26** We leave here open the case of non-bidirectional nP systems, which can also be simulated by hP systems, but require additional costs, in terms of steps and object transfers.

## 6 Planar representation of hP Systems

Classical tree-based P systems allow a “nice” planar representation, where the parent/child relationships between membranes are represented by Venn-like diagrams. Can we extend this representation to cover our dag-based hP systems?

In this section we will show that any hP system structurally based on a canonical dag can still be *intensionally* represented by hierarchically nested planar regions, delimited by Jordan curves (simple closed curves). Conversely, we also show that any set of hierarchically nested planar regions delimited by Jordan curves can be interpreted as a canonical dag, which can form the structural basis of a number of hP systems.

We will first show how to represent a canonical dag as a set of hierarchically nested planar regions.

### Algorithm 27 (Algorithm for visually representing a canonical dag)

Without loss of generality, we consider a canonical dag  $(V, \delta)$  of order  $n$ , where vertices are indexed according to an arbitrary topological order implied by the arcs, by considering parents before the children, i.e.,  $V = \{v_i \mid 1 \leq i \leq n\}$ , where  $(v_i, v_j) \in \delta$  implies  $i < j$ . Figure 13 shows side by side a simple height 1 canonical dag and its corresponding hypergraph representation. Note the *intensional* representation (as opposed to the *extensional* one):  $v_2$  is not totally included in  $v_1$ , although all vertices included in  $v_2$ , i.e.,  $v_4$  and  $v_5$ , are also included in  $v_1$ . A possible topological order is  $v_1, v_2, v_3, v_4, v_5$ .

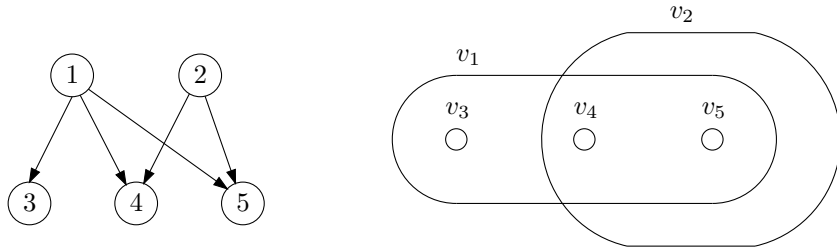


Figure 13: A simple canonical dag and its corresponding hypergraph representation.

For each vertex  $v_i$ , we associate a distance  $\psi_i = \frac{1}{2^{(n-i+1)}}$ , for  $i \in \{1, \dots, n\}$ . For Figure 13,  $\psi_i = \frac{1}{32}, \frac{1}{16}, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}$ , for  $i \in \{1, \dots, n\}$ .

We process the vertices in reverse topological order  $v_n, \dots, v_1$ , at each step  $i$  representing the current vertex  $v_i$  by a planar region  $R_i$ .

First, we set parallel horizontal axis  $X_o$  and  $X_p$ , vertically separated by distance  $3(n-1)$ . Secondly, we set points  $o_1, \dots, o_n$  on  $X_o$ , such that  $o_i$  and  $o_{i+1}$  are separated by distance 3, for  $1 \leq i \leq n-1$ . We define  $o_i$  as the *origin point* of  $v_i$ , and write  $o_i = \text{origin}(v_i)$ . Finally, we set points  $p_1, \dots, p_n$  on  $X_p$ , such that  $p_i$  and  $p_{i+1}$  are separated by distance 3, for  $1 \leq i \leq n-1$ . We define  $p_i$  as the *corridor point* of  $v_i$ .

Figure 14 shows the construction of  $X_o, X_p, o_i$  and  $p_i$ , for the dag of Figure 13, where  $n = 5$ .

If the current vertex  $v_i$  is a sink, then  $R_i$  is a circle with radius  $\frac{1}{2}$  centered at  $o_i$ .

If the current vertex  $v_i$  is a non-sink, then  $R_i$  is constructed as follows: Assume that the children of  $v_i$  are  $w_1, \dots, w_{n_i}$ , and their (already created) regions are  $S_1, \dots, S_{n_i}$ .

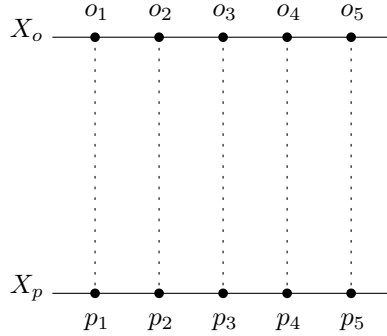


Figure 14: Construction of  $X_o, X_p, o_i$  and  $p_i$ , for the dag of Figure 13, where  $n = 5$ .

Consider line segments  $l_0, l_1, \dots, l_{n_i}$ , where  $l_0$  is bounded by  $o_i$  and  $p_i$ , and  $l_j$  is bounded by  $p_i$  and  $origin(w_j)$ , for  $j \in \{1, \dots, n_i\}$ . Let  $L_0, L_1, \dots, L_{n_i}, T_1, \dots, T_{n_i}$  be the regions enclosed by Jordan curves around  $l_0, l_1, \dots, l_{n_i}, S_1, \dots, S_{n_i}$ , at a distance  $\psi_i$ , and let  $R'_i = L_0 \cup \bigcup_{j=1, \dots, n_i} L_j \cup \bigcup_{j=1, \dots, n_i} T_j$ . We define  $R_i$  as the external contour of  $R'_i$ . This definition will discard all internal holes (such as the dashed enclosed regions of Figure 16), if any, without introducing any additional containment relations between our regions. The details of our construction guarantee that no internal hole will ever contain an origin point.  $\square$

Figure 15 shows an intermediate step (left) and the final step (right) of applying Algorithm 27 on Figure 13. The representation of Figure 15 is topologically equivalent to the hypergraph representation of Figure 13 (right). Figure 16 shows the side by side, another dag and its corresponding planar region representation; internal holes are represented by dotted lines. Our objective here was not to create “nice” visualisations, but to prove that it is possible to represent an arbitrary canonical dag, i.e., an arbitrary hP system structurally based on a canonical dag, by hierarchically nested planar regions.

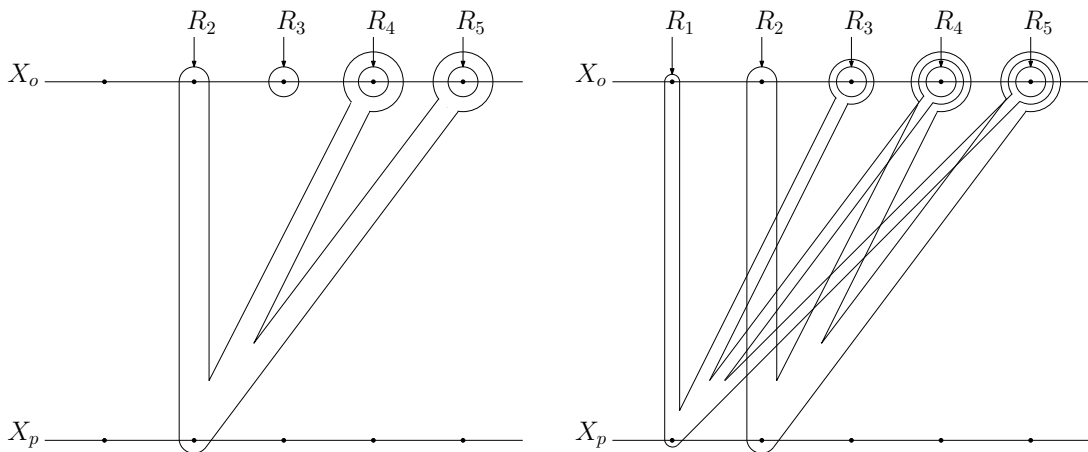


Figure 15: An intermediate step and the final step of applying Algorithm 27 on Figure 13.

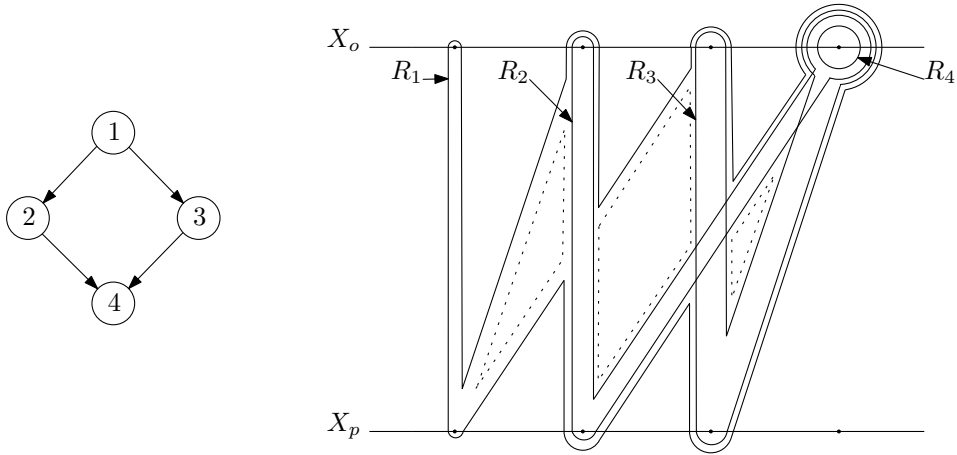


Figure 16: A height two dag and its corresponding representation, built by Algorithm 27.

We will next show that, for any finite set of hierarchically nested planar regions, we can build a corresponding canonical dag (i.e., the underlying structure of an hP system).

**Algorithm 28 (Algorithm for building a canonical dag from finite set of hierarchically nested planar regions)**

Assume that we have  $n$  hierarchically nested planar regions,

1. Label each planar region by  $R_i$ ,  $i \in \{1, \dots, n\}$ ,
2. If  $R_i$  directly nests  $R_j$  then draw an arc from a vertex  $v_i$  to a vertex  $v_j$ ,  $i, j \in \{1, \dots, n\}$ ,  $i \neq j$ .

□

We now show that a canonical graph produced from Algorithm 28 does not contain any cycles. Our proof is by contradiction. Let us assume a directed graph  $G$  produced from Algorithm 28 contains a cycle  $v_i, \dots, v_k, \dots, v_i$ . Then every vertex in a cycle has an incoming arc. If vertex  $v_k$  is a maximal element in a cycle, with respect to direct nesting, then its corresponding planar region  $R_k$  have the largest region area among planar regions in a cycle. Since no other planar region in a cycle can contain  $R_k$ , there are no arc incident to vertex  $v_k$ . Hence, there is no cycle in  $G$ .

**Remark 29** We leave open the problem of representing dags (that is hP systems) that contain transitive arcs.

## 7 Summary

We have proposed a new model, as an extension of P systems, that provides a better communication structure and we believe is often more convenient for modelling real-world

applications based on tree structures augmented with secondary or shared communication channels.

We have shown that hP systems functionally extends the basic functionality of transition P systems and neural P systems, even though the underlying structure of hP systems is different. In the dag/hypergraph model of hP systems we can have a natural separation of computing cells (sinks) from communication cells (hyperedges). This model also allows us to easily represent multiple inheritance and/or to distribute computational results (as specified by a dag) amongst several different parts of a membrane structure.

We note that the operational behavior of hP systems is separate from the topological structure of a membrane system. In this paper, we illustrated hP systems using the computational rules of nP systems, where multisets of objects are repeatedly changed within cells, by using a fixed set of multiset rewriting rules, or transferred between cells, using several possible transfer modes.

Finally, we provided a intuitive visualisation of hP systems, by showing that any set of hierarchically nested planar regions (which represents any set of cells ordered by containment) is equivalent to, or modelled by, a dag without transitive arcs. We provided simple algorithms to translate between these two interpretations.

Part B of this paper will explore this model further and support it with a more extensive set of examples.

## References

- [1] C. Berge: *Hypergraphs: Combinatorics of Finite Sets*. Elsevier Science Publishers (1989)
- [2] C. Carathéodory: *Theory of Functions of a Complex Variable*. Vol. 1, Chelsea Publishing Company (1954)
- [3] G. Ciobanu: Distributed Algorithms Over Communicating Membrane Systems. *Bio Systems*, 70, 2, 123–133 (2003)
- [4] W. F. Doolittle: Uprooting the Tree of Life. *Scientific American* 282, 90–95 (2000)
- [5] T. -O. Ishdorj, M. Ionescu: Replicative-Distribution Rules in P Systems with Active Membranes. *Proceeding of the First International Colloquium on Theoretical Aspects of Computing (ICTAC 2004)*, 68–83 (2004)
- [6] C. Li: Master Thesis. The University of Auckland, Supervisor: R. Nicolescu (2008)
- [7] C. Martín-Vide, G. Păun, J. Pazos, A. Rodríguez-Patón: Tissue P Systems. *Theoretical Computer Science*, 296, 2, 295–326 (2003)
- [8] G. Păun: Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1, 108–143 (2000) (and Turku Center for Computer Science, TUCS Report 208, November 1998)



- [9] G. Păun: *Membrane Computing—An Introduction*. Springer-Verlag (2002)
- [10] G. Păun: *Introduction to Membrane Computing*. Proceeding of the First Brainstorming Workshop on Uncertainty in Membrane Computing, 17–65 (2004)
- [11] G. Păun, R. A. Păun: Membrane Computing as a Framework for Modeling Economic Processes. *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005)*, 11–18 (2005)
- [12] M. Slikker, A. Van Den Nouweland: *Social and Economic Networks in Cooperative Game Theory*. Kluwer Academic Publishers (2001)
- [13] V. I. Voloshin: *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications*. American Mathematical Society (2002)
- [14] C. Zandron, C. Ferretti, G. Mauri: Solving NP-Complete Problems Using P Systems with Active Membranes. *Proceeding of the Second International Conference on Unconventional Models of Computation*, 289–301 (2000)