Int. J. of Computers, Communications & Control, ISSN 1841-9836, E-ISSN 1841-9844
Vol. V (2010), No. 2, pp. 224-237

Towards Structured Modelling with Hyperdag P Systems

R. Nicolescu, M.J. Dinneen, Y.-B. Kim

Michael J. Dinneen, Yun-Bum Kim and Radu Nicolescu
Department of Computer Science, University of Auckland,
Private Bag 92019, Auckland, New Zealand
E-mail: {radu,mjd,yun}@cs.auckland.ac.nz

Abstract: Although P systems are computationally complete, many real world models, such as socio-economic systems, databases, operating systems and distributed systems, seem to require more expressive power than provided by tree structures. Many such systems have a primary tree-like structure augmented 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 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.

Keywords: hyperdag P systems, tissue and 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 [10]. 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 visualized as a Venn diagram, where nesting denotes a parent–child relationship. For example, Figure 1 [12] shows the same P system structure with nine membranes, labelled as $1, \ldots, 9$, both as a rooted tree and as a Venn diagram.

More, recently, neural P systems [11], abbreviated as nP systems (also known as tissue P systems [7]), have been introduced, partially to overcome the limitations of the tree model. Essentially, these systems organize their cells in an arbitrary graph. For example, ignoring for the moment the actual contents of cells (states, objects, rules), Figure 2 illustrates the membrane structure of a simple nP system, consisting of three cells, $\sigma_1, \sigma_2, \sigma_3$, where cell σ_1 is the output cell.

A large variety of rules have been used to describe the operational behavior of P systems, such as 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 rules operate by communicating immutable objects and P systems with active membranes combine all three rule types. For a comprehensive overview and more details, we refer the reader to [11, 12].

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 [16],

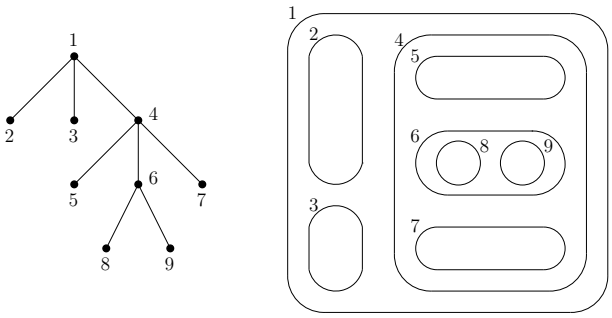


Figure 1: A P system structure represented as a tree and as a Venn diagram.

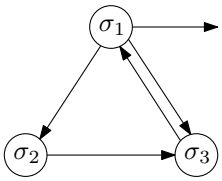


Figure 2: An nP system structure represented as a digraph.

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 visualization, 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 a 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 factor [13, 14, 15].

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. Models based on general graphs, such as nP systems, while allowing any direct communications, 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 modularization of concerns, practical parallel implementation could be less efficient, if the locality of reference is not enforced, and visualizations 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 3, (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 nP systems. Despite using an apparently

restricted structure, we show in Section 4 that our dag-based model has the same computational power as graph-based tissue P systems 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, such as systems based on symport/antiport rules.

We support our view with a realistic example (see Examples 11 and 12), inspired from computer networking, modelled as an hP system with a shared communication channel (broadcast channel).

Classical P systems allow a "nice" planar visualization, where the parent–child relationships between membranes are represented by Venn-like diagrams. We show in Section 5 that the extended membrane structure of hP systems can still be visualized by hierarchically nested planar regions.

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 $x_1,\ldots,x_n$, such that $\{(x_1,x_2),\ldots,(x_{n-1},x_n)\} \subseteq A$. A *cycle* is a path $x_1,\ldots,x_n$, 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*, $A(A^{-1}(x))\backslash\{x\} = A(A^{-1}(\{x\}))\backslash\{x\}$ 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 $x_1,\ldots,x_n$, 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. Example dags are shown in Figures 3 and 4.

We consider a variant hypergraph definition, based on multisets, as an extension of 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 3 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.

Generalizing the above hypergraph definition, a height n *generalized hypergraph* is a system (X,E), recursively built via a sequence of n hypergraphs $(X_1,E_1),\ldots,(X_n,E_n)$ where $X_1 = X$, $X_i \cap E_i = \emptyset, X_{i+1} =$

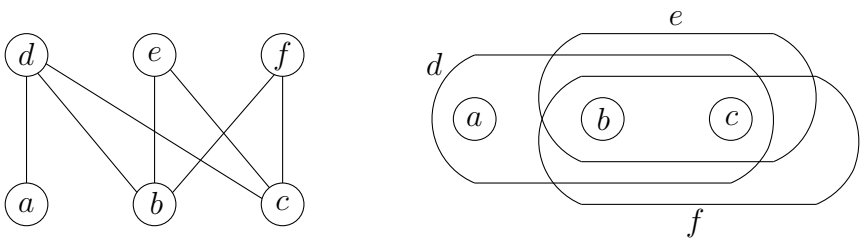


Figure 3: A simple height 1 dag and its corresponding hypergraph representation.

$X_i \cup E_i$, $e \cap E_i \neq \emptyset$ for $\forall e \in E_{i+1}$ and $E = \bigcup_{i \in \{1,\ldots,n\}} E_i$. 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 $i \in \{1,\ldots,n\}$.

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 4 shows a height 2 dag and its corresponding height 2 hypergraph (X,E), where $X = X_1 = \{a,b,c,d,e\}$, $E_1 = \{f,g,h\}$, $E_2 = \{i\}$, $E = \{f,g,h,i\}$.

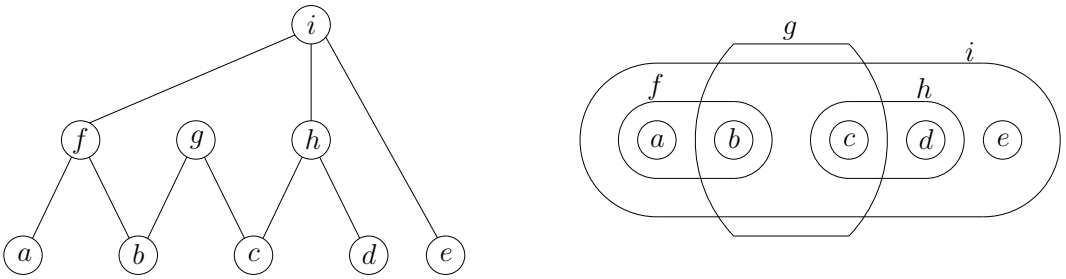


Figure 4: A height 2 dag and its corresponding height 2 hypergraph.

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 $o_1^{n_1} \ldots o_k^{n_k}$, where $o_i \in O$, $n_i \geq 0$, and, in the canonical form, letters appear in sorted order, i.e., $o_1 < \cdots < o_k$, 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 λ represents the empty multiset, and string concatenation represents multiset union, e.g., $(a^2c) \cdot (ab) = a^3bc$.

3 Hyperdag P Systems

In this paper we use the definition of nP systems, as given in [11], that coincides with an early definition of tissue P systems as given in [7]. Our definition includes a small technical correction (slight ambiguity). For details, please see our technical report [8].

As in the mentioned nP systems definition, we will use 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: $|_O, |_{go}, |_{out} : O_{tot}^* \to O^*$, by $o|_O = o, o_{go}|_{go} = o, o_{out}|_{out} = o$ for $o \in O$, and otherwise λ. For example, $a^2a_{go}^3b^4b_{go}|_{go} = a^3b$.

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 system object transfer modes. In fact, we could also discard the *go* tag, as it corresponds to the union of these new target 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 or via tags) are possible, and indeed seem natural, but this is beyond the scope of this article.

We will now define hP systems, as an apparent restriction of nP systems, where the underlying structure is a dag, with several other adjustments.

Definition 1 (Hyperdag P systems) An *hP system* (of degree m) is a system $\Pi = (O, \sigma_1, \ldots, \sigma_m, \delta, I_{out})$, where:

1. O is an ordered finite non-empty alphabet of *objects*;
2. $\sigma_1, \ldots, \sigma_m$ 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 $i \in \{1, \ldots, m\} \backslash I_{out}$,
3. δ is a set of dag parent–child arcs on $\{1, \ldots, m\}$, i.e., $\delta \subseteq \{1, \ldots, m\} \times \{1, \ldots, m\}$, representing *duplex* communication channels between cells;
4. $I_{out} \subseteq \{1, \ldots, m\}$ 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 used in neural P systems by a more structured arc set δ (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 systems.

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 [11]. However, we could adopt and adapt any other rule set, from other variants or extensions of P systems, such as rewriting, symport/antiport or boundary rules [12].

Definitions of *configurations*, *transitions*, *computations* and *results of computations* in hP systems are similar to definitions used for nP systems (see also [8]), with the following essential additions and differences, here informally stated:

- The *rewriting mode* α and *transfer mode* β may not be fixed from the start, i.e., they may vary, for each cell σ_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 5 schematically shows the possible object transfers from a cell σ_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*).

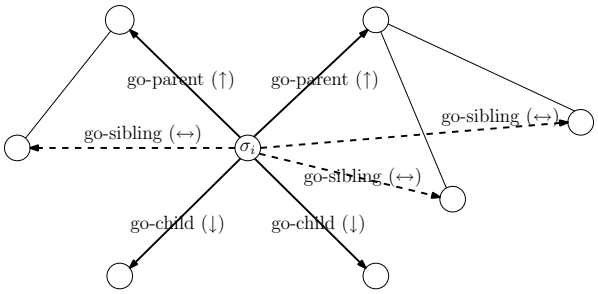


Figure 5: An annotated hP system indicating possible transfers from cell σ_i. The parent-child axis is top-down. Plain lines indicate parent-child relations and dashed lines indicate siblings. Arrows at the end of long thick lines, plain or dashed, indicate possible transfer directions from cell σ_i.

More formal definitions follow.

Definition 2 (Configurations) A *configuration* of the hP system Π is an m-tuple of the form $(s_1w_1,\ldots,s_mw_m)$, with $s_i \in Q_i$ and $w_i \in O^*$, for $1 \leq i \leq m$. The m-tuple $(s_{1,0}w_{1,0},\ldots,s_{m,0}w_{m,0})$ is the *initial configuration* of Π.

Definition 3 (Rewriting and transfer modes) For an hP system of degree m,

- the *object rewriting mode* is a function $\alpha : \bigcup_{i\in\{1,\ldots,m\}} \{i\} \times Q_i \rightarrow \{min, par, max\}$.
- the *object transfer mode* is a function $\beta : \bigcup_{i\in\{1,\ldots,m\}} \{i\} \times Q_i \rightarrow \{repl, one, spread\}$.

Definition 4 (Rewriting steps) For each cell σ_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 $sw_1 \rightarrow s'w'_1, \ldots, sw_k \rightarrow s'w'_k \in P_i$, $k \geq 1$, such that $w_1 \ldots w_k \subseteq x, y = (x-w_1 \ldots w_k) \cup w'_1 \ldots w'_k$, and there is no $sw \rightarrow s'w' \in P_i$, such that $w_1 \ldots w_k w \subseteq x$ (note that rules can be combined only if they start from the same state s and end in the same state s').

Definition 5 (Transition steps) Given two configurations $C_1 = (s_1 w_1, \ldots, s_m w_m)$ and $C_2 = (s'_1 w''_1, \ldots, s'_m w''_m)$, we write $C_1 \Rightarrow_{\alpha,\beta} C_2$, for α and β (as defined in Definition 3), if the conditions below are met.

First, we apply rewriting steps (as defined in Definition 4) 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 $z^{\uparrow}_{j,k}$, $z^{\downarrow}_{j,k}$, $z^{\leftrightarrow}_{j,k}$, the outgoing multisets from j to k, where $j \in \{1,\ldots,m\}$ and, respectively, $k \in \delta^{-1}(j)$, $k \in \delta(j)$, $k \in \delta(\delta^{-1}(j))\backslash\{j\}$:

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

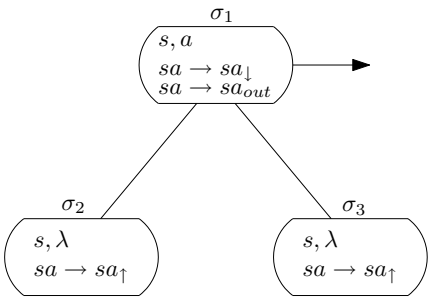
Finally, we set $w''_i = w'_i|_O \cup \bigcup_{j\in\delta^{-1}(i)} z^{\uparrow}_{j,i} \cup \bigcup_{j\in\delta(i)} z^{\downarrow}_{j,i} \cup \bigcup_{j\in\delta(\delta^{-1}(i))\backslash\{i\}} z^{\leftrightarrow}_{j,i}$, for $i \in \{1,\ldots,m\}$.

Definition 6 (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. Within the family of P systems, two systems are *functionally equivalent*, if they yield the same computational results.

Example 7 Consider two hP systems, Π_1 and Π_2, (which are functionally equivalent).

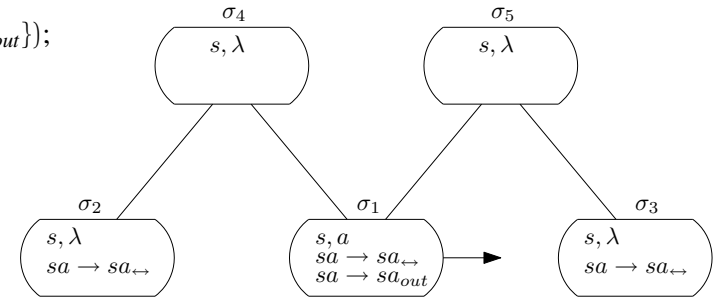
$\Pi_1 = (O, \sigma_1, \sigma_2, \sigma_3, \delta, I_{out})$, where:

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



$\Pi_2 = (O, \sigma_1, \sigma_2, \sigma_3, \sigma_4, \sigma_5, \delta, I_{out})$, where:

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



4 Relations between P Systems, Neural P Systems and Hyperdag P Systems

Theorem 8 (Hyperdag P systems include non-dissolving transition P systems).
Any non-dissolving[1] 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 Π_T [12], we build a functionally equivalent hP system Π_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.
$\Pi_T = (O, C, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, i_o)$, $\Pi_H = f(\Pi_T) = (O', \sigma'_1, \ldots, \sigma'_m, \delta, I_{out})$.

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

[1] A dissolving membrane occurs if we allow rules that tell a membrane to disappear where its remaining objects go to its parent membrane; see [12].

- $\delta = \mu$;
- $I_{out} = \{i_o\}$;
- The object rewriting mode is the *max* constant function, i.e., $\alpha(i,s) = max$, for $i \in \{1,\ldots,m\}, s \in Q_i$;
- The object transfer mode is the *spread* constant function, i.e., $\beta(i,s) = spread$, for $i \in \{1,\ldots,m\}, s \in Q_i$.

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

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

Remark 9 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 here omit such extensions.

Proving that hP systems also simulate 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 10 The combination of *go-sibling* ($\leftrightarrow$) with *repl* object transfer mode enables 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 count the number of messages. In hP systems, a local broadcast needs only one message to siblings, while graph- or digraph-based systems need $n-1$ messages.

Example 11 Figure 6 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 the same 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 12 Figure 7 shows the computer network of Figure 6, modelled as a graph (if we omit arrows) or as a digraph (if we consider the arrows). Note that the graph- or digraph-based models, such as nP systems, 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".

We can now mention an important theorem comparing hP systems and nP systems.

Theorem 13 (Hyperdag P systems can simulate symmetric neural P systems).
Any symmetric nP system can be simulated by an hP system, with the same number of steps and object transfers.

Proof: The simulation details are given in our research report [8]. □

Remark 14 We leave here open the case of non-symmetric nP systems, which can also be simulated by hP systems, but require additional costs, in terms of steps and object transfers.

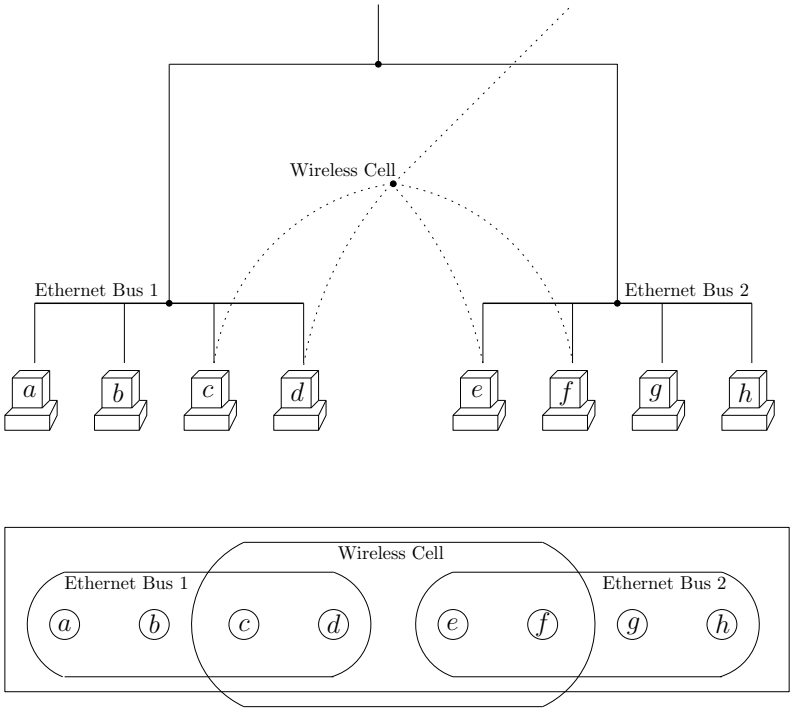


Figure 6: A computer network and its corresponding hypergraph representation.

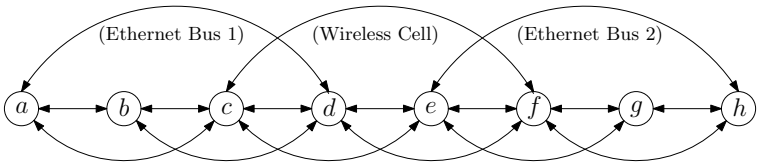


Figure 7: The digraph representation of the computer network of Figure 6.

5 Planar representation of hyperdag P 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 15 (Algorithm for visually representing a canonical dag)

Without loss of generality, we consider a canonical dag (V, δ) 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 8 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), where 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.

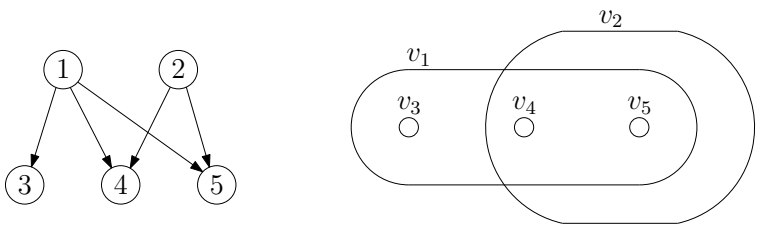


Figure 8: A simple canonical dag and its corresponding hypergraph representation.

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

We process the vertices in reverse topological order $v_n,\ldots,v_1$, 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 $o_1,\ldots,o_n$ 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 $o_i = origin(v_i)$. Finally, we set points $p_1,\ldots,p_n$ 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 9 shows the construction of X_o, X_p, o_i and p_i, for the dag of Figure 8, where $n = 5$.

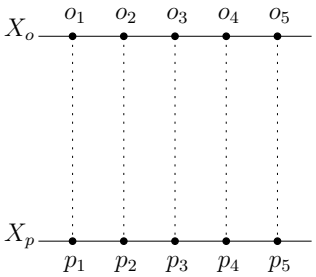


Figure 9: Construction of X_o, X_p, o_i and p_i, for the dag of Figure 8, where $n = 5$.

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 $w_1,\ldots,w_{n_i}$, and their (already created) regions are $S_1,\ldots,S_{n_i}$. Consider line segments $l_0, l_1,\ldots,l_{n_i}$, where l_0 is bounded by o_i and p_i, and l_j is bounded by p_i and $origin(w_j)$, for $j \in \{1,\ldots,n_i\}$. Let L_0, $L_1,\ldots,L_{n_i}$, $T_1,\ldots,T_{n_i}$ be the regions enclosed by Jordan curves around l_0, $l_1,\ldots,l_{n_i}$, $S_1,\ldots,S_{n_i}$, at a distance ψ_i, and let $R'_i = L_0 \cup \bigcup_{j=1,\ldots,n_i} L_j \cup \bigcup_{j=1,\ldots,n_i} T_j$. 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 10), 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. □

Figure 10 shows the side by side, a dag and its corresponding planar region representation; internal holes are represented by dotted lines. Our objective here was not to create "nice" visualizations, 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.

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

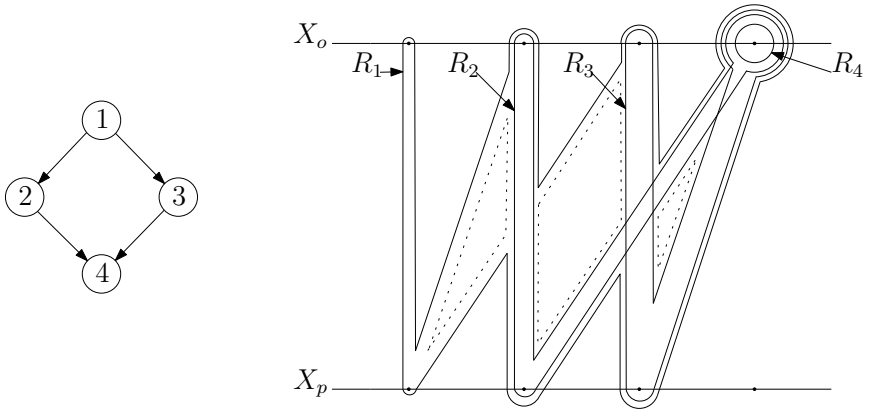


Figure 10: A height 2 dag and its corresponding representation, built by Algorithm 15.

Assume that we have n hierarchically nested planar regions,

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

□

We now show that a canonical digraph produced from Algorithm 16 does not contain any cycles. Our proof is by contradiction. Let us assume a digraph G produced from Algorithm 16 contains a cycle $v_i, \ldots, v_k, \ldots, v_i$. 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 17 We present in [9] a solution to the problem of representing dags (that contain transitive arcs) by a set of simple regions, where direct containment denotes a parent–child relation.

6 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 nP systems, even though the underlying structure of hP systems is different. In dag-based 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 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 an intuitive visualization 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.

This paper is part of an ongoing research dedicated to structured modelling and model checking of P systems.

Dedication

This article is dedicated to Mario J. Pérez-Jiménez, on the occasion of his 60th birthday (November 2008).

Bibliography

[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(2):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, *Validating P System as Distributed Computing Models*, Master thesis, 2008.
[7] C. Martín-Vide, G. Păun, J. Pazos, A. Rodríguez-Patón, Tissue P Systems, *Theoretical Computer Science*, 296(2):295–326, 2003.
[8] R. Nicolescu, M. J. Dinneen, Y.-B. Kim, Structured Modelling with Hyperdag P Systems: Part A, CDMTCS Research Report Series, *CDMTCS-342*, 1–24, December 2008. `http://www.cs.auckland.ac.nz/CDMTCS/researchreports/342hyperdagA.pdf`
[9] R. Nicolescu, M. J. Dinneen, Y.-B. Kim, Discovering the Membrane Topology of Hyperdag P Systems, *Proceeding of the Tenth Workshop on Membrane Computing (WMC10 2009)*, Curtea de Argeş, Romania, August 24–27, 426–451, 2009.
[10] 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).
[11] G. Păun, *Membrane Computing: An Introduction*. Springer-Verlag, 2002.
[12] G. Păun, Introduction to Membrane Computing, *Proceeding of the First Brainstorming Workshop on Uncertainty in Membrane Computing*, 17–65, 2004.
[13] 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.
[14] M. Slikker, A. Van Den Nouweland, *Social and Economic Networks in Cooperative Game Theory*, Kluwer Academic Publishers, 2001.
[15] V. I. Voloshin, *Coloring Mixed Hypergraphs: Theory, Algorithms and Applications*, American Mathematical Society, 2002.
[16] 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.

Michael J. Dinneen received his PhD in 1996 from the University of Victoria in Canada. He is currently a Senior Lecturer at the University of Auckland. His research interests are in combinatorial algorithms, graph algorithms, uncoventional computing models and network design.

Yun-Bum Kim is currently a PhD student at the University of Auckland, New Zealand. In 2007 he has completed a MSc thesis in network optimization design problems and currently has interests in molecular and distributed computing.

Radu Nicolescu (PhD Bucharest, MACM, MemIEEE) is currently a Senior Lecturer at the University of Auckland, New Zealand. He has research interests in formal languages, information complexity and service oriented computing.