Hierarchical automata and P-systems

N. Sabadini [1,2]

Dipartimento di Scienze CC. FF. MM.,
Università degli Studi dell'Insubria,
Como, Italy

R.F.C. Walters [3]

Dipartimento di Scienze CC. FF. MM.,
Università degli Studi dell'Insubria,
Como, Italy

Abstract

In the context of a compositional theory of automata [8] (CP automata) we define a notion of hierarchical automaton, and show how the P-systems of Gh. Păun [9] may be described in terms of these automata. A distributive law for CP automata is proved which relates two different views of hierarchical systems: the first view is that hierarchical systems consist of layers, with communication between adjacent layers; the second is that a hierarchical system is an evolving hierarchy or tree structure.

1 Introduction.

There are a wide variety of models of hierarchical systems, from the statecharts of Harel [4], to the mobile ambients of Cardelli and Gordon [2], to the recently defined P-systems of Gh. Păun [9]. In this paper we study the model of Păun. The aim of the paper is to define a notion of *hierarchical automaton*, in terms of a subalgebra of the algebra of CP automata introduced in [8], and to show how P-systems can be described in this algebra.

This comparison suggests two new aspects of the algebra of CP automata. Firstly, in this paper the parallel operation of CP automata is used to model

[1] We thank Piergiulio Katis for helpful discussions. The research has been financially supported by the Dipartimento di Scienze Chimiche, Fisiche e Matematiche of the University of Insubria, Como, Italy, and by the Italian Progetto Cofinanziato MURST *Metamodelli Computazionali* (COMETA).

[2] Email: nicoletta.sabadini@uninsubria.it

[3] Email: robert.walters@uninsubria.it

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

the interaction of two adjacent layers of a hierarchy, whereas in previous papers ([5], [6], [8]) it has been used to model interaction of processes in a single layer. As a result, in this paper a system (an expression in the algebra of CP automata) has a layer structure, or equivalently, in the paradigm of P-systems a membrane structure. Secondly we prove a new distributive law for CP automata which provides a precise relation between two fundamentally different ways of looking at hierarchical systems. Such a system may be thought of (i) as a vertical composition of layers, each layer having its own evolution in communication with adjacent layers, or alternatively (ii) as an evolving tree structure - evolving snapshots of the hierarchical structure.

P-systems were introduced as a class of distributed parallel computing devices of a biochemical type. The basic model consists of a membrane structure composed by several cell-membranes, hierarchically embedded in a main membrane called the skin membrane. The membranes delimit regions and can contain objects, which evolve according to given evolution rules associated with the regions. Such rules are applied in a maximally parallel manner: at each step, all the objects which can evolve should evolve. Membranes themselves may be dissolved. A computation device is obtained: we start from an initial configuration and we let the system evolve. A computation halts when no further rule can be applied. The objects in a specified output membrane (or expelled through the skin membrane) are the result of the computation. Many variants are considered in [9], [10], [11] and [1]. A survey and an up-to-date bibliography can be found at the web address http://bioinformatics.bio.disco.unimib.it/psystems.

The theory of P-systems as it stands is non-compositional. A connection has been made with automata theory in [3] though their automata are also non-compositional. Connection between P-systems and the ambient calculus of Cardelli and Gordon (which arose as a calculus to describe secure and mobile administrative domains in the internet) have been made in [12].

The motivation for introducing CP automata in the first place was to define an algebra of finite automata (for describing control), with operations being parallel composition (called here *product* for non-communicating parallel, and restricted product for communicating parallel), and sequential (what we call here the *sum*). Associated there are also parallel and sequential feedback operations. In order to define the operations it was necessary to add structure to the automata (*interfaces* for parallel composition, and *conditions* for sequential composition). Although we were interested initially in finite automata, one can also consider recursive equations in the algebra [8]. With these it is possible to express unbounded reconfigurable automata, and hence, for example, Turing machines. This paper shows how hierarchical structures may be expressed in the algebra of CP automata and introduces a fundamental relation between the parallel and sequential operations - the distributive law. In further work we will investigate other hierarchical models such as statecharts, ambient calculus and hierarchical Turing machines.

2 CP automata

The algebra we use later to describe hierarchical systems has as elements certain automata, which we call *case-place automata* (or CP automata) but which have also been called *cospans of spans of graphs* from their mathematical origin. Much of this section is described in [8], but here we have changed some names and emphasis to suit the application to hierarchical systems.

We denote the set of vertices of a graph $\mathbf{G}$ as $vert(\mathbf{G})$, and the set of arcs as $arc(\mathbf{G})$.

Definition 2.1 A *CP automaton* $\mathcal{G}$ consists of a graph $\mathbf{G}$, four sets X, Y, A, B and four functions

$$\partial_0 : arc(\mathbf{G}) \to X,\ \partial_1 : arc(\mathbf{G}) \to Y,$$
$$\gamma_0 : A \to vert(\mathbf{G}),\ \gamma_1 : B \to vert(\mathbf{G}).$$

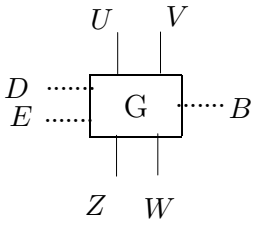
A *behaviour* of $\mathcal{G}$ is a path in the graph $\mathbf{G}$.

The graph $\mathbf{G}$ is called the *centre* of the automaton. We remark that ∂_0, ∂_1 may be thought of as *labellings* of the arcs of $\mathbf{G}$ in the alphabets X, Y, respectively. These labellings will be used in the restricted product of two CP automata, the operation which expresses communicating parallel processes. Alternatively, one may think of the vertices and arcs of $\mathbf{G}$ as the *states* and *transitions* of the system, whereas the elements of X, Y are the transitions of the interfaces. We call X the *top interface* of the automaton, and Y the *bottom interface* - automata communicate through these interfaces. (Note that in [8] we used the words "left" and "right" instead of top and bottom for the interfaces - the change is to fit in with the intuition of hierarchy.) Often the interface sets will be products of sets; for example the top interface of $\mathbf{G}$ may be $X = U \times V$, and the bottom interface may be $Y = Z \times W$, and we then speak of U and V as the top interfaces, and Z and W as the bottom interfaces. If we ignore the functions γ_0, γ_1 a CP automaton is a (particular type of) *span of graphs* as defined in [5], where the reader may find more details and examples.

The set A represents a *condition* on the states in which the automaton may come into existence, and the set B a condition in which it may cease to exist. We call A the *in-condition* of the automaton, and B the *out-condition*. The functions γ_0, γ_1 of a CP automaton will be used in the restricted sum of CP automata - an operation which expresses change of configuration of processes. The meaning of the in- and out-conditions will become clearer in the section on the restricted sum of automata. Often the condition sets will be sums of sets; for example the in-condition may be $A = D + E$, and we then speak of D and E as in-conditions.

There is a useful informal graphical representation of CP automata (ignoring the conditions this is described in [5]). For example, we will represent a CP automaton with top interface $U \times V$, bottom interface $Z \times W$, in-condition

$D + E$, and out-condition B, by a diagram of the following sort:



For simplicity we use the same names ∂_0, ∂_1, γ_0, γ_1 for the four functions of *any* CP automaton where there is no risk of confusion, introducing further suffixes when clarification is needed. We use symbols $X, Y, Z, U, V, W...$ for the (top and bottom) interfaces, and symbols $A, B, C, D, E, F, I, ...$ for the (in- and out-) conditions.

Example 2.2 An example of a CP automaton is provided by a Mealy automaton with input alphabet X, output alphabet Y, state set S, transition function $\delta : X \times S \to S \times Y$, initial state s_0 and final states F. The corresponding CP automaton has top interface X, bottom interface Y; $vert(\mathbf{G}) = S$, an arc from $s_1 \in S$ to $s_2 \in S$ labelled on the top by x and on the bottom by y if $\delta(x, s_1) = (s_2, y)$; incondition $A = \{s_0\}$ and outcondition $B = F$. The functions γ_0, γ_1 are inclusions. The reader is warned that this example gives a false impression of the strength of the model we are describing. It is essential, for example, in expressing the changing geometry of a system that the functions γ_0, γ_1 not be restricted solely to inclusions. The sets A and B are not to be thought of as initial and final states, but rather as conditions under which a change of geometry might occur. Another difference with Mealy automata is that the sets X and Y need not be input and output but rather interfaces on which communication occurs with connected components.

To describe the distributive law in a later section we will need the notion of *isomorphism* of CP automata.

Definition 2.3 Given two CP automata $\mathcal{G} =(\mathbf{G}, X, Y, A, B, \partial_{0,\mathcal{G}}, \partial_{1,\mathcal{G}}, \gamma_{0,\mathcal{G}}, \gamma_{1,\mathcal{G}})$ and $\mathcal{H} =(\mathbf{H}, X, Y, A, B, \partial_{0,\mathcal{H}}, \partial_{1,\mathcal{H}}, \gamma_{0,\mathcal{H}}, \gamma_{1,\mathcal{H}})$ an isomorphism from $\mathcal{G}$ to $\mathcal{H}$ is a graph isomorphism $\varphi : \mathbf{G} \to \mathbf{H}$ such that $\partial_{0,\mathcal{H}} \circ \varphi = \partial_{0,\mathcal{G}}$, $\partial_{1,\mathcal{H}} \circ \varphi = \partial_{1,\mathcal{G}}$, $\varphi \circ \gamma_{0,\mathcal{G}} = \gamma_{0,\mathcal{H}}$, $\varphi \circ \gamma_{1,\mathcal{G}} = \gamma_{1,\mathcal{H}}$.

2.1 Operations

2.1.1 Parallel composition

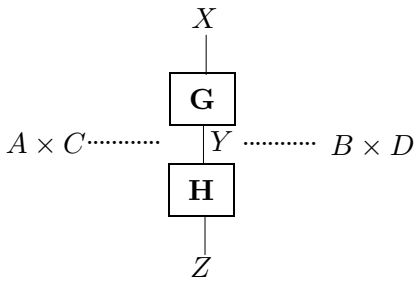
Definition 2.4 Given two CP automata

$$\mathcal{G} = (\mathbf{G}, X, Y, A, B, \partial_0, \partial_1, \gamma_0, \gamma_1),$$
$$\mathcal{H} = (\mathbf{H}, Y, Z, C, D, \partial_0, \partial_1, \gamma_0, \gamma_1)$$

the *restricted product* (communicating parallel composition) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} \cdot \mathcal{H}$ is the CP automaton whose set of vertices is $vert(\mathbf{G}) \times vert(\mathbf{H})$ and whose set of arcs is that subset of $arc(\mathbf{G}) \times arc(\mathbf{H})$ consisting of pairs of arcs (g, h) such that $\partial_1(g) = \partial_0(h)$. The interfaces and conditions of $\mathcal{G} \cdot \mathcal{H}$ are $X, Z, A \times C, B \times D$, and the four functions are

$$\partial_{0,\mathcal{G}\cdot\mathcal{H}}(g,h) = \partial_{0,\mathcal{G}}(g),\ \partial_{1,\mathcal{G}\cdot\mathcal{H}}(g,h) = \partial_{1,\mathcal{H}}(h),$$
$$\gamma_{0,\mathcal{G}\cdot\mathcal{H}} = \gamma_{0,\mathcal{G}} \times \gamma_{0,\mathcal{H}}\ ,\ \gamma_{1,\mathcal{G}\cdot\mathcal{H}} = \gamma_{1,\mathcal{G}} \times \gamma_{1,\mathcal{H}}\ .$$

Diagrammatically we represent the restricted product as follows:



Closely related to the restricted product is the free product of CP automata.

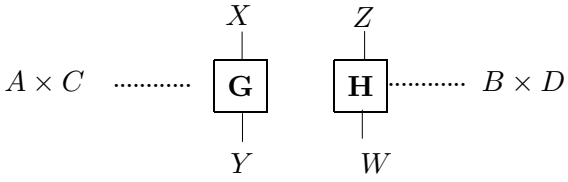
Definition 2.5 Given two CP automata

$$\mathcal{G} = (\mathbf{G}, X, Y, A, B, \partial_0, \partial_1, \gamma_0, \gamma_1),$$
$$\mathcal{H} = (\mathbf{H}, Z, W, C, D, \partial_0, \partial_1, \gamma_0, \gamma_1)$$

the *free product* (parallel composition with no communication) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} \times \mathcal{H}$ is the CP automaton whose set of vertices is $vert(\mathbf{G}) \times vert(\mathbf{H})$ and whose set of arcs is $arc(\mathbf{G}) \times arc(\mathbf{H})$. The interfaces and conditions of $\mathcal{G} \times \mathcal{H}$ are $X \times Z, Y \times W, A \times C, B \times D$, and the four functions are

$$\partial_{0,\mathcal{G}\times\mathcal{H}} = \partial_{0,\mathcal{G}} \times \partial_{0,\mathcal{H}},\ \partial_{1,\mathcal{G}\times\mathcal{H}} = \partial_{1,\mathcal{G}} \times \partial_{1,\mathcal{H}},$$
$$\gamma_{0,\mathcal{G}\times\mathcal{H}} = \gamma_{0,\mathcal{G}} \times \gamma_{0,\mathcal{H}}\ ,\ \gamma_{1,\mathcal{G}\times\mathcal{H}} = \gamma_{1,\mathcal{G}} \times \gamma_{1,\mathcal{H}}\ .$$

Diagrammatically we represent the free product as follows:



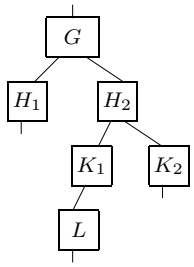
If we ignore the functions γ_0 , γ_1 of the CP automata the restricted product of CP automata is the *span composition* of [5] and the free product is the *tensor product* of the corresponding spans of graphs. For some examples of how these operations may be used to model concurrent systems see that paper. From a circuit theory point of view these operations correspond, respectively, to the series and parallel operations of circuit components.

Definition 2.6 An *elementary automaton* is a CP automaton with top interface a single set X and bottom interface a product of sets $Y_1 \times Y_2 \times \cdots \times Y_n$.

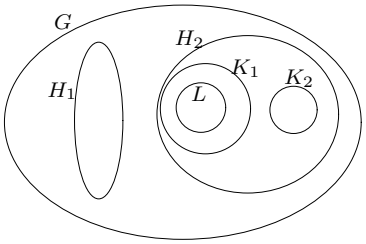
Remark 2.7 An expression of elementary automata using only the operations free and restricted product has, diagrammatically, a tree structure, and it is exactly this tree structure we use to model a hierarchy. For example, the expression

$$\mathcal{G}\bullet(\mathcal{H}_1 \times (\mathcal{H}_2 \bullet ((\mathcal{K}_1 \bullet \mathcal{L}) \times \mathcal{K}_2)))$$

is represented diagrammatically as



An alternative way of picturing such expressions, adapted to the biological intuition of Păun, is in terms of membranes, as follows:



Definition 2.8 Given two CP automata

$$\mathcal{G} = (\mathbf{G}, X, Y, A, B, \partial_0, \partial_1, \gamma_0, \gamma_1),$$
$$\mathcal{H} = (\mathbf{H}, X, Y, B, C, \partial_0, \partial_1, \gamma_0, \gamma_1)$$

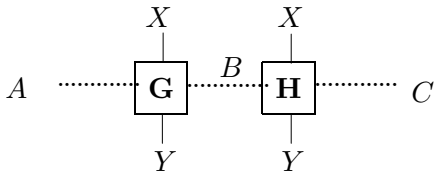
the *restricted sum* (change of configuration) of $\mathcal{G}$ and $\mathcal{H}$, denoted $\mathcal{G} + \mathcal{H}$ is the CP automaton whose set of arcs is $arc(\mathbf{G}) + arc(\mathbf{H})$ and whose set of vertices is $(vert(\mathbf{G}) + vert(\mathbf{H}))/\sim$; that is $vert(\mathbf{G}) + vert(\mathbf{H})$ quotiented by

the relation $\gamma_{1,\mathcal{G}}(b) \sim \gamma_{0,\mathcal{H}}(b)$ (for all $b \in B$). The interfaces and conditions of $\mathcal{G} + \mathcal{H}$ are X, Y, A and C, and the four functions are

$$\begin{aligned} \partial_{0,\mathcal{G}+\mathcal{H}} &= (\partial_{0,\mathcal{G}} \,|\, \partial_{0,\mathcal{H}}),\ \partial_{1,\mathcal{G}+\mathcal{H}} = (\partial_{1,\mathcal{G}} \,|\, \partial_{1,\mathcal{H}}), \\ \gamma_{0,\mathcal{G}+\mathcal{H}} &= in_{G_0} \circ \gamma_{0,\mathcal{G}},\ \ \gamma_{1,\mathcal{G}+\mathcal{H}} = in_{H_0} \circ \gamma_{1,\mathcal{H}}\,. \end{aligned}$$

A behaviour of $\mathcal{G} + \mathcal{H}$ is initially a behaviour of $\mathcal{G}$, and then, if a state in the image of B is reached, the behaviour may change to a behaviour of $\mathcal{H}$. The intended interpretation is that initially only the process $\mathcal{G}$ exists; when a state in B is reached the process $\mathcal{G}$ may die and the process $\mathcal{H}$ be created. Alternatively, the process $\mathcal{G}$ may evolve in state B into the process $\mathcal{H}$.

The diagrammatic representation of the restricted sum is as follows:



2.1.2 Sum feedback

Definition 2.9 Given a CP automaton

$$\mathcal{G} = (\mathbf{G}, X, Y, A + B, C + B, \partial_0, \partial_1, \gamma_0, \gamma_1),$$

the *sum feedback* of $\mathcal{G}$ with respect to $\mathcal{B}$, denoted $\mathsf{Sfb}_B(\mathcal{G})$ is the CP automaton whose set of arcs is $arc(\mathbf{G})$ and whose set of vertices is $vert(\mathbf{G})/\sim$; that is $vert(\mathbf{G})$ quotiented by the relation $(\gamma_1 \circ in_B)(b) \sim (\gamma_0 \circ in_B)(b)$ (for all $b \in B$). The interfaces and conditions of $\mathsf{Sfb}_B(\mathcal{G})$ are X, Y, A and C, and the four functions are defined as follows:

$$\begin{aligned} \partial_{0,\mathsf{Sfb}_B(\mathcal{G})} &= \partial_{0,\mathcal{G}},\ \partial_{1,\mathsf{Sfb}_B(\mathcal{G})} = \partial_{1,\mathcal{G}}, \\ \gamma_{0,\mathsf{Sfb}_B(\mathcal{G})} &= \gamma_{0,\mathcal{G}} \circ in_A,\ \ \gamma_{1,\mathsf{Sfb}_B(\mathcal{G})} = \gamma_{1,\mathcal{G}} \circ in_C. \end{aligned}$$

The diagrammatic representation of $\mathsf{Sfb}_B(\mathcal{G})$ involves joining the out-condition B to the in-condition B.

Remark 2.10 In [8] a further operation is described, namely *product feedback*. In describing the hierarchical structure of a system this operation is not necessary.

2.1.3 Adjusting the conditions

Notice that any function f from A to B may be regarded as a CP automaton, for any choice of top and bottom interface X and Y, in the following way: the

graph has no arcs, its vertex set is B, the in-condition is $f : A \to B$ and the out-condition is $1_B : B \to B$. Restricted sum of such *functional* CP automata is exactly normal functional composition. There are also CP automata which are the *reverse* of functions. Function $f : A \to B$ gives a reverse functional automaton with vertex set B in-condition $1_B : B \to B$, and out-condition $f : A \to B$.

Given two CP automata we wish to compose, it may happen that the conditions are not appropriate to allow the composition, but that some modification is necessary. To this end we may adjust the in-conditions by composition with functional automata, and composition of out-conditions with reverse functional automata. It is useful to have a special notation for this. If $\mathcal{G}$ is a CP automaton with in-condition A and out-condition B it is useful to indicate these conditions by writing $\mathcal{G}$ as $\mathcal{G}_B^A$. Then the result of composing $\mathcal{G}$ with the functional CP $f : A^{'} \to A$ and the reverse functional CP automaton $g : B^{'} \to B$ is indicated by $\mathcal{G}_{g:B^{'}\to B}^{f:A^{'}\to A}$ or even $\mathcal{G}_{B^{'}}^{A^{'}}$ when the functions f, g are clear from the context. In general an adjustment of in-condition we denote by adding a superscript, and adjusting an out-condition by adding a subscript.

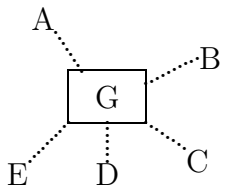
Remark 2.11 Notice that the restricted sum seems very close to sequential composition. However this is deceptive; consider the parallel composite $\mathcal{G} \times \mathcal{H}$ of two processes, where $\mathcal{G}$ has out-condition T (a one element terminal state), and $\mathcal{H}$ has in- and out-condition $vert(H)$. Notice that $\mathcal{G} \times \mathcal{H}$ has out-condition $T \times vert(H)$ which may be adjusted by composing with the isomorphism $T \times vert(H) \cong vert(H)$. The interpretation of the restricted sum $(\mathcal{G} \times \mathcal{H})_{vert(H)} + \mathcal{H}$ is that the process $\mathcal{G}$ dies upon reaching its terminal state leaving the process $\mathcal{H}$ still running.

3 Distributive laws

In [8] we described some distributive laws between product operations and sum operations. These laws however are not sufficient for the purposes of this paper. We describe now a generalization (it derives from an analogous law for slightly different operations described in [7]). Notice that the sum operations we describe are not the + of Milner's CCS for which, famously, there is not a distributive law. The distributive laws we are about to describe come from looking at CP automata in a different way - we will for the moment ignore the separation of conditions into top and bottom conditions.

Remark 3.1 For this section we will make a slight change to the notion of CP automaton. Instead of having an in-condition and an out-condition we will allow a *family of conditions* To obtain a CP automaton in the old sense it simply suffices to make an assignment of which are the in- and which are the out- conditions. For fixed top and bottom interfaces X and Y we represent a CP automaton $\mathcal{G}$, with conditions A, B, C, D, E say, as a box with five dotted

lines emerging in different directions:



This is a more geometrical (wysiwig) than algebraic view of CP automata. We will now define a geometric analogue of the restricted sum of CP automata.

Definition 3.2 A *family of CP automata*, $\mathbb{G}$,with top interface X and bottom interface Y, consists of a reflexive graph R (the *indexing* graph of the family) and

(i) for each vertex $r \in R$ a CP automaton $\mathbb{G}_r$ (with the same interfaces X and Y);

(ii) for each edge $\rho : r_1 \to r_2$ two conditions, $\mathbb{G}_{1,\rho} : A \to vert(\mathbb{G}_{r_1})$, a condition of $\mathbb{G}_{r_1}$ and $\mathbb{G}_{2,\rho} : A \to vert(\mathbb{G}_{r_2},)$ a condition of $\mathbb{G}_{r_2}$, with the same domain A;

(iii) for the reflexive edges $\epsilon : r \to r$, $\mathbb{G}_{1,\epsilon} = \mathbb{G}_{2,\epsilon} = 1_{vert(\mathbb{G}_r)} : vert(\mathbb{G}_r) \to vert(\mathbb{G}_r)$.

We now define a new operation on CP automata, which involves glueing together the individual automata of the family using the conditions of the family.

Definition 3.3 The *restricted sum of a family* of CP automata $\mathbb{G}$ is the CP automata denoted $\sum \mathbb{G}$ (or with abuse of notation $\sum_{r \in R} \mathbb{G}_r$) and formed as follows:

(i) the set of edges of $\sum \mathbb{G}$ is the disjoint union of the edge sets of $\mathbb{G}_r$ $(r \in R)$;

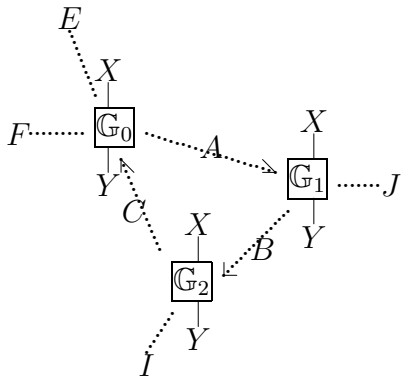
(ii) the labelling of the edges is the same as that of the edges in $\mathbb{G}_r$ $(r \in R)$;

(iii) the vertex set of $\sum \mathbb{G}$ is the quotient of the disjoint union of the vertex sets of $\mathbb{G}_r$ $(r \in R)$ by an equivalence relation; the equivalence relation is the smallest one such for each $\rho : r_1 \to r_2$ in R, $\mathbb{G}_{1,\rho}(a)$ is equivalent to $\mathbb{G}_{2,\rho}(a)$ for each a in the domain of $\mathbb{G}_{1,\rho}$ and $\mathbb{G}_{2,\rho}$;

(iv) the conditions of $\sum \mathbb{G}$ are the conditions of the $\mathbb{G}_r$ $(r \in R)$ not occurring as $\mathbb{G}_{i,\rho}$ $(i = 1, 2; \rho$ an edge of $R)$.

Remark 3.4 Just as we have used a diagrammatic representation for the earlier operations on CP automata, we will use a suggestive diagrammatic representation for the restricted sum of a family of CP automata. The representation should be clear from the following example. Let R be the reflexive

graph with vertices r_0, r_1, r_2 and (non-reflexive) edges being $\rho_0 : r_0 \to r_1, \rho_1 : r_1 \to r_2, \rho_2 : r_2 \to r_0$. The restricted sum of the R indexed family $\mathbb{G}$ is denoted:



Notice that for simplicity we have omitted the reflexive edges (and will do so throughout the paper). The sets A, B, C are the domains of the conditions of $\mathbb{G}_{i,\rho_0}, \mathbb{G}_{i,\rho_1}, \mathbb{G}_{i,\rho_2}$ respectively ($i = 1, 2$). The sets E, F, I, J are conditions of the restricted sum $\sum \mathbb{G}$.

Example 3.5 The two previously defined sum operations of CP automata are special cases of this new restricted sum. The restricted sum of previous section is the case where the indexing graph R of the family of CP automata has two vertices, and one non-reflexive edge from one vertex to the other. The sum feedback operation is the special case in which the graph R has one vertex and one non-reflexive edge. The restricted sum of families of CP automata is not really more powerful than restricted sum and sum feedback together: one can show that in the presence of certain constants a restricted sum of a family may be expressed in terms of the two simpler operations.

Definition 3.6 Given two families of CP automata, $\mathbb{G}$ with top interface X and bottom interface Y and with indexing graph R, and $\mathbb{H}$ with top interface Z and bottom interface W and with indexing graph S, the *free product* of $\mathbb{G}$ and $\mathbb{H}$ is the family, denoted $\mathbb{G} \times \mathbb{H}$, with top interface $X \times Z$, bottom interface $Y \times W$, indexing graph $R \times S$ and defined as follows: $(\mathbb{G} \times \mathbb{H})_{(r,s)} = \mathbb{G}_r \times \mathbb{H}_s$ and $(\mathbb{G} \times \mathbb{H})_{i,(\rho,\sigma)} = \mathbb{G}_{i,\rho} \times \mathbb{H}_{i,\sigma}$ for $r, \rho \in R$, $s, \sigma \in S$ and $i = 1, 2$.

Definition 3.7 Given two families of CP automata, $\mathbb{G}$ with top interface X and bottom interface Y, $\mathbb{H}$ with top interface Y and bottom interface Z - and with indexing graphs R, S respectively the *restricted product* of $\mathbb{G}$ and $\mathbb{H}$ is the family, denoted $\mathbb{G} \cdot \mathbb{H}$, with top interface X, bottom interface Z, indexing graph $R \times S$ and defined as follows: $(\mathbb{G} \cdot \mathbb{H})_{(r,s)} = \mathbb{G}_r \cdot \mathbb{H}_s$ and $(\mathbb{G} \cdot \mathbb{H})_{i,(\rho,\sigma)} = \mathbb{G}_{i,\rho} \times \mathbb{H}_{i,\sigma}$ for $r, \rho \in R$, $s, \sigma \in S$ and $i = 1, 2$.

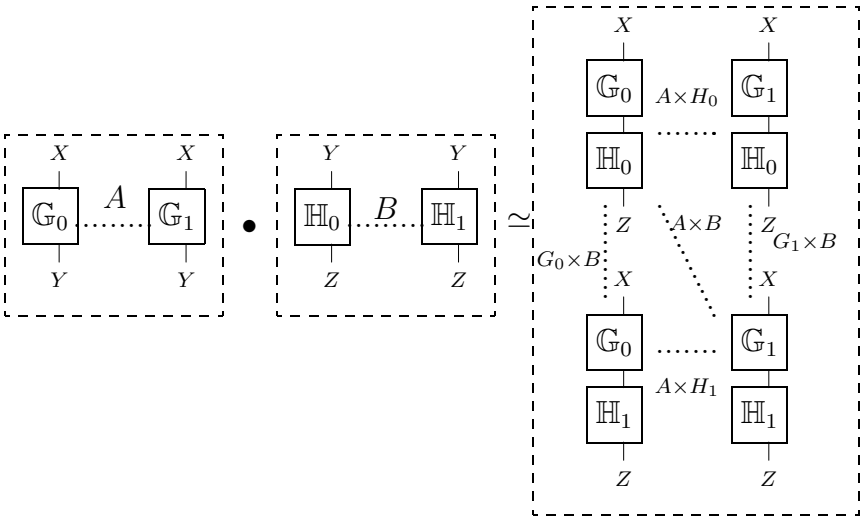
Proposition 3.8 *(The Distributive Laws)*

(i) *Given two families of CP automata,* $\mathbb{G}$ *with top interface* X *and bottom interface* Y*, and* $\mathbb{H}$ *with top interface* Z *and bottom interface* W*, then* $(\sum \mathbb{G}) \times (\sum \mathbb{H}) \simeq \sum(\mathbb{G} \times \mathbb{H})$;

(ii) *Given two families of CP automata,* $\mathbb{G}$ *with top interface* X *and bottom interface* Y*,* $\mathbb{H}$ *with top interface* Y *and bottom interface* Z *- then* $(\sum \mathbb{G}) \cdot (\sum \mathbb{H}) \simeq \sum(\mathbb{G} \cdot \mathbb{H})$.

Proof. We omit the proof which, though detailed, is routine. □

Example 3.9 Consider the reflexive graph R with two vertices $0, 1$, two reflexive edges $\epsilon : 0 \to 0$, $\epsilon : 1 \to a1$ and one non-reflexive edge $\rho : 0 \to 1$. Then $R \times R$ has 4 vertices $(0.0), (0, 1), (1, 0), (1, 1)$ and 5 non-reflexive edges $(\epsilon, \rho) : (0, 0) \to (0, 1), (\epsilon, \rho) : (1, 0) \to (1, 1), (\rho, \epsilon) : (0, 0) \to (1, 0), (\rho, \epsilon) : (0, 1) \to (1, 1), (\rho, \rho) : (0, 0) \to (1, 1)$. Consider two families $\mathbb{G}, \mathbb{H}$ both indexed by R. Then the second distributive law diagrammatically is:



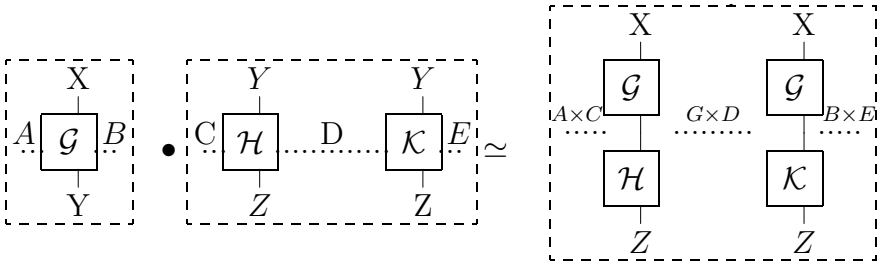
where G_i, H_i are the vertex sets of $\mathbb{G}_i, \mathbb{H}_i$ respectively.

Example 3.10 The distributive laws given in [8] are special cases of these two laws. For example, the law

$$\mathcal{G}_B^A \cdot (\mathcal{H}_D^C + \mathcal{K}_E^D) \cong \mathcal{G}_{vert(\mathcal{G})}^A \cdot \mathcal{H}_D^C + \mathcal{G}_B^{vert(\mathcal{G})} \cdot \mathcal{K}_E^D,$$

arises from considering family $\mathbb{G}$ with indexing graph having only one vertex and no reflexive edges, and family $\mathbb{H}$ having indexing graph with two vertices,

and one non-reflexive edge from one vertex to the other. Diagrammatically



where $G = vert(\mathcal{G})$. Notice the meaning of this law: automaton $\mathcal{H}$ may evolve into $\mathcal{K}$ under condition D, hence $\mathcal{G} \bullet \mathcal{H}$ may evolve into $\mathcal{G} \bullet \mathcal{K}$ in *any state of* $\mathcal{G}$, coupled with a D state of $\mathcal{H}$. A similar interpretation may be given to the previous example which describes the condition under which the communicating parallel composition of two evolving processes may evolve.

4 Hierarchical automata

Definition 4.1 An *elementary automaton* (already defined above) is a CP automaton with top interface a single set X and bottom interface a product of sets $Y_1 \times Y_2 \times \cdots \times Y_n$. A *product-type automaton* is an expression of elementary automata formed using only the free and restricted product operations. A *hierarchical automaton* is an expression in the algebra of CP automata built from elementary automata using the operations restricted product, free product, and restricted sum of families.

Proposition 4.2 *Any hierarchical automaton is isomorphic to a restricted sum of a family of product-type automata.*

Proof. Expand the expression using the distributive laws. This is the precise analogy of the fact that arithmetic expressions may expanded into sums of monomials.

Remark 4.3 The view of hierarchical systems as a vertical composition of layers, each layer having its own evolution in communication with adjacent layers, corresponds to the fact that a hierarchical automaton *may* be a restricted product of sums of free products of elementary automata - each factor in the restricted product is a *layer*. The above proposition shows that such a system may also be viewed as a restricted sum of product-type automata - that is as an evolving tree structure - evolving snapshots of the hierarchical structure.

□

5 Hierarchical automata and P-systems

The aim of this section is to show how the principal features of Păun's P-systems [9] can be modelled with hierarchical automata. In reality there is not a single notion of P-system - Paŭn speaks of "small jungle of variants" created by introducing different biological concepts. However all variants include at least the following features:

(i) A P-system has an initial tree structure of *membranes* - the tree structure may change during evolution. These membranes delimit *regions*.

(ii) Each region contains a state at any moment (a *set of molecules*) and, particular to the region, there are possible transitions of this state which are local - that is the transitions are not affected by and do not affect other regions.

(iii) For each region there may be transitions which change the state in the region, but also change the state of one of the regions contained in this region (a so-called *in* instruction).

(iv) For each region there may be state transitions which change the state and the state of the region containing the given region (a so-called *out* instruction).

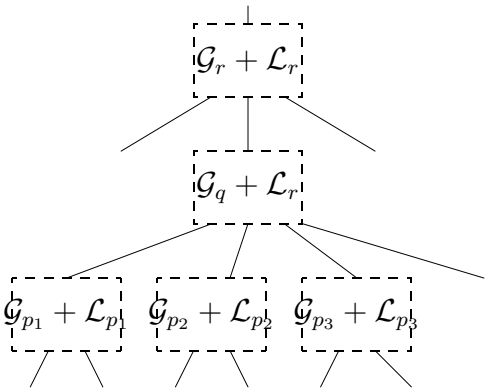
(v) For each region there may be, in a given state, a transition which *dissolves* the membrane containing the region, changing the state of the region outside (a δ instruction).

For a more detailed discussion see the papers on P-systems cited in the introduction. For our purposes, these five points are sufficient - we will explain how each of these five features may be modelled.

Firstly each region p of a P-system is a restricted sum $\mathcal{G}_p + \mathcal{L}_p$ of two elementary automata , one $\mathcal{G}_p$ corresponding to the case that the membrane is intact, the other $\mathcal{L}_p$ to the case that the membrane is dissolved. When the membrane is intact the states of a region are exactly those as given in the definition of P-systems. When the membrane is dissolved the region has precisely one state. A number of regions $p_1, p_2, \cdots, p_n$ whose outer membranes are contained in a single membrane together are modelled as the free product $(\mathcal{G}_{p_1}+\mathcal{L}_{p_1})\times(\mathcal{G}_{p_2}+\mathcal{L}_{p_2})\times\cdots\times(\mathcal{G}_{p_n}+\mathcal{L}_{p_n})$. Finally if q is the region immediately outside the regions $p_1, p_2, \cdots, p_n$ then the system consisting of all the regions $q, p_1, p_2, \cdots, p_n$ is modelled by

$$(\mathcal{G}_q + \mathcal{L}_q) \bullet ((\mathcal{G}_{p_1} + \mathcal{L}_{p_1}) \times (\mathcal{G}_{p_2} + \mathcal{L}_{p_2}) \times \cdots \times (\mathcal{G}_{p_n} + \mathcal{L}_{p_n})).$$

A diagram should make this clearer (r is the region immediately outside q):



It is now clear how the first feature of P-systems, the initial tree structure of membranes, is modelled by hierarchical automata. We have now to discuss the transitions and their labelling. To model the fact that there are entirely local transitions (feature 2) we need that every state has an idling transition, labelled by an idling action ϵ. This means that any transition in a region, q say, also labelled on all interfaces by ϵ may occur in the total system without reference to other connected regions (in this case $r, p_1, p_2, \cdots, p_n$). Such transitions are the local transitions. Transitions corresponding to *out* instructions in region q are pairs of transitions, one of $\mathcal{G}_q$ and one of $\mathcal{G}_r$, labelled with ϵ on all but the common interface between regions r and q. On the common interface the two transitions have the same label. Transitions corresponding to *in* instructions of region q are modelled similarly. Finally, let us see by a very simple example how dissolving membranes is handled. Consider the hierarchical automata $\mathcal{H} \bullet (\mathcal{G} + \mathcal{L}) \bullet \mathcal{K}$ where $\mathcal{H} = \mathcal{G}' + \mathcal{L}'$ and $\mathcal{K} = \mathcal{G}'' + \mathcal{L}''$, .and where the top and bottom interfaces of $\mathcal{G} + \mathcal{L}$ are the same. Using the distributive laws

$$\mathcal{H} \bullet (\mathcal{G} + \mathcal{L}) \bullet \mathcal{K} \simeq \mathcal{H} \bullet \mathcal{G} \bullet \mathcal{K} + \mathcal{H} \bullet \mathcal{L} \bullet \mathcal{K}.$$

We have not as yet specified the transitions of the automaton $\mathcal{L}$. It has one transition for each label; the transition is labelled on both interfaces with the corresponding label. It is easy then to see that $\mathcal{H} \bullet \mathcal{L} \bullet \mathcal{K} \simeq \mathcal{H} \bullet \mathcal{K}$. This reflects the fact that the state and internal rules of $\mathcal{G} + \mathcal{L}$ have disappeared. The interpretation is that the system evolves from a hierarchy of three membranes $\mathcal{H} \bullet \mathcal{G} \bullet \mathcal{K}$ to one in which there are only two membranes $\mathcal{H} \bullet \mathcal{K}$. The situation in which the top and bottom interfaces of $\mathcal{G} + \mathcal{L}$ are not the same - for example, the bottom interface breaks into two - is more complex because a choice is involved. Arbitration may be necessary between the two bottom interfaces, and so $\mathcal{H} \bullet \mathcal{L}$ may be considered as $\mathcal{H}$ with a modified bottom interface.

References

[1] Besozzi Daniela, Claudio Zandron, Giancarlo Mauri, and Nicoletta Sabadini, *P Systems with Gemmation of Mobile Membranes,* Proceedings ICTCS 2000, Lecture Notes in Computer Science, Springer Verlag, **2202**, (2001), 136-153.

[2] Cardelli L., and A. Gordon, *Mobile ambients*, in Proceedings of FoSSaCS'98 (M. Nivat, ed.), Lecture Notes in Computer Science, Springer Verlag, **1378** (1998), 140–155.

[3] Csuhaj-Varjú Erzsébet, and György Vaszil, *P automata*, preprint.

[4] Harel David,. *Statecharts: A Visual Formalism for Complex Systems,.* Science of Computer Programming, **8**(3) (1987), 231–274.

[5] Katis, N. Sabadini, and R.F.C. Walters, *Span(Graph): A categorical algebra of transition systems*, Proceedings Algebraic Methodology and Software Technology, Lecture Notes in Computer Science, Springer Verlag, **1349** (1997), 307–321.

[6] Katis P., N. Sabadini, R.F.C. Walters, *On the algebra of systems with feedback and boundary*, Rendiconti del Circolo Matematico di Palermo Serie II, Suppl. **63** (2000), 123–156.

[7] Katis P., N. Sabadini, R.F.C. Walters, An algebra of automata for reconfiguring networks of interacting components, Poster at CT2000.

[8] Katis P., N. Sabadini, and R.F.C: Walters; *A formalization of the IWIM Model.* in: Proc. COORDINATION 2000,(Eds.) Porto A., Roman G.-C., Lecture Notes in Computer Science, Springer Verlag, **1906** (2000), 267–283.

[9] Păun Gh., *Computing with membranes*, Journal of Computer and System Sciences, **61**, 1 (2000), 108–143.

[10] Păun Gh., *Computing with membranes A variant: P systems with polarized membranes*, Intern. J. of Foundations of Computer Science, **11**, 1 (2000), 167–182.

[11] Păun Gh., G. Rozenberg, and A. Salomaa, *Membrane computing with external output*, Fundamenta Informaticae, **41**, 3 (2000), 259–266.

[12] Petre I., and L. Petre, *Mobile ambients and P systems*, Workshop on Formal Languages, FCT99, Iaşi, 1999, J. Universal Computer Sci., **5**, 9 (1999), 588–598.