ELSEVIER

Theoretical Computer Science 296 (2003) 295–326

Theoretical Computer Science

www.elsevier.com/locate/tcs

Tissue P systems

Carlos Martín-Vide[a,*], Gheorghe Păun[b,a,1], Juan Pazos[c], Alfonso Rodríguez-Patón[c]

[a] *Research Group on Mathematical Linguistics, Rovira i Virgili University, Pl. Imperial Tàrraco 1, 43005 Tarragona, Spain*

[b] *Institute of Mathematics of the Romanian Academy, PO Box 1-764, 70700 Bucureşti, Romania*

[c] *Department of Artificial Intelligence, Faculty of Computer Science, Polytechnical University of Madrid, Campus de Montegancedo, Boadilla del Monte 28660, Madrid, Spain*

Abstract

Starting from the way the inter-cellular communication takes place by means of protein channels (and also from the standard knowledge about neuron functioning), we propose a computing model called a *tissue P system*, which processes symbols in a multiset rewriting sense, in a net of cells. Each cell has a finite state memory, processes multisets of symbol-impulses, and can send impulses ("excitations") to the neighboring cells. Such cell nets are shown to be rather powerful: they can simulate a Turing machine even when using a small number of cells, each of them having a small number of states. Moreover, in the case when each cell works in the maximal manner and it can excite all the cells to which it can send impulses, then one can easily solve the Hamiltonian Path Problem in linear time. A new characterization of the Parikh images of ET0L languages is also obtained in this framework. Besides such basic results, the paper provides a series of suggestions for further research.
© 2002 Elsevier Science B.V. All rights reserved.

Keywords: Natural computing; Membrane computing; P systems; Chomsky hierarchy; Lindenmayer hierarchy; NP-complete problems

1. Introduction

This paper has two starting points: the inter-cellular communication (of chemicals, energy, information) by means of complex networks of protein channels (we will elaborate about this topic in the next section), and the way the neurons co-operate,

* Corresponding author.

[1] Work supported by a grant of NATO Science Committee, Spain, 2000–2001, by the Programa Cátedra of Fundación Banco Bilbao Vizcaya Argentaria (BBVA), and by Facultad de Informática, Universidad Politécnica de Madrid.

processing impulses in the complex net established by synapses (we discuss some details in Section 3).

The common mathematical model of these two kinds of symbol-processing mechanisms is the net of finite state devices, and this is the type of computing mechanisms we are going to consider: networks of finite-automata-like processors, dealing with symbols, according to local states (available in a finite number for each "cell"), communicating through these symbols, along channels ("axons") specified in advance.

Note that the neuron modeling was the very starting point of the theory of finite automata [22,18], that symbol processing neural networks have a rich (and controversial) history (see [5] and its references), and that networks of string-processing finite automata have appeared in many contexts ([8,14,16,21], etc.), but our models are different in many respects from all these previous models.

Having in mind the bio-chemical reality we refer to, a basic problem concerns the organization of the bunch of symbols available in each node (cell in a tissue or neuron), and the easiest and most natural answer is: no organization. Formally, this means that we have to consider *multisets* of symbols, sets with multiplicities associated with their elements, but without any ordering among symbols. In this way, we need a kind of finite automata dealing with multisets of symbols, a topic which falls into an area of (theoretical) computer science not very much developed, although some recent (see, e.g., [19,11]), or not so recent (see, e.g., [4]) approaches can be found in the literature. Actually, most of the vivid area of membrane computing [25] is devoted to multiset processing (see details at the web address `http://psystems.disco.unimib.it`.).

The computing models we propose here, under the name of *tissue P systems*, in short, *tP systems*, consist of several *cells*, related by protein channels. In order to preserve also the neural intuition, we will use the shorter and suggestive name of *synapses* for these channels. Each cell has a state from a given finite set and can process multisets of *objects* (chemical compounds in the case of cells, impulses in the case of the brain), represented by symbols from a given alphabet. The standard rules are of the form $sM \to s'M'$, where s, s' are states and M, M' are multisets of symbols. Some of the elements of M' may be marked with the indication "go", and this means that they have to immediately leave the cell and pass to the cells to which we have direct links through synapses. This communication (transfer of symbol-objects) can be done in a replicative manner (the same symbol is sent to all adjacent cells), or in a non-replicative manner; in the second case we can send all the symbols to only one neighboring cell, or we can distribute them, non-deterministically. One more choice appears in using the rules $sM \to s'M'$: we can apply such a rule only to one occurrence of M (that is, in a sequential, *minimal* way), or to all possible occurrences of M (a *parallel* way), or, moreover, we can apply a maximal package of rules of the form $sM_i \to s'M'_i, 1 \leqslant i \leqslant k$, that is, involving the same states s, s', which can be applied to the current multiset (the *maximal* mode).

By the combination of the three modes of processing objects and the three modes of communication among cells, we get nine possible behaviors of our machinery.

Now, the problem arises how to use such a device as a computing one. One possibility is to start from a given initial configuration (that is, initial states of cells and initial multisets of symbol-objects placed in them) and to let the system proceed until reaching a halting configuration, where no further rule can be applied, and to associate a result with this configuration. Because of the non-deterministic behavior of a tP system, starting from one given initial configuration we can reach arbitrarily many different halting configurations, hence we can get arbitrarily many outputs. Another possibility is to also provide *inputs*, at various times of a computation, and to look for the outputs related to them. Here we will consider only the first possibility, of *generative* tP systems, and the output will be defined by sending symbols out of the system. To this aim, one cell will be designated as the output one, and in its rules $sM \to s'M'$ we will also allow that symbols from M' are marked with the indication "out"; such a symbol will immediately leave the system, contributing to the result of the computation.

At the first sight, such a machinery (a finite net of finite state devices) seems not to be very powerful, for instance, as compared with the Turing machines. Thus, it is rather surprising to find that tP systems with a small number of cells (two or four), each of them using a small number of states (at most five or four, respectively) can simulate any Turing machine, even in the non-cooperative case, that is, only using rules of the form $sM \to s'M'$ with M being a singleton (containing only one copy of only one symbol); moreover, this is true for all modes of communication for the minimal mode of using the rules, and, in the cooperative case, also when using the parallel or the maximal mode of processing the objects.

In the case when the rules are non-cooperative and we use them in the maximal mode, a characterization of the Parikh images of ET0L languages is obtained, which completes the study of the computing power of our devices (showing that in the *parallel* and *maximal* cases we dot not get the computational universality).

The above mentioned results indicate that our cells are "very powerful" (maybe, from a biological point of view, "too powerful"); as their power lies in using states, hence in remembering their previous work, a natural idea is to consider tP systems with a low bound on the number of states in each cell. In view of the previously mentioned results, tP systems with at most 1, 2, 3, or 4 states per cell are of interest. We only briefly consider this question here, and we show that even reduced tP systems as those which use only one state in each cell can be useful: using such a net we can solve the Hamiltonian Path Problem in linear time (this is a direct consequence of the structure of a tP system, of the maximal mode of processing objects, and of the power of replicating the objects sent to all adjacent cells); remember that HPP is an NP-complete problem.

The power of tP systems with a reduced number of states per component remains to be further investigated—many other natural research topics will be also mentioned in the last section of the paper. The richness of these ideas for further investigations (as well as the results mentioned above) is a proof that these tissue-like counterparts of the "classic" P systems deserve a systematic examination.

2. The inter-cellular communication

The P systems are computing models inspired from the structure and the functioning of the living cells—see [25,7], or the above mentioned web page.

In short, a P system consists of a membrane structure (a three-dimensional structure of vesicles, all of them placed in a main vesicle, delimited by the "skin membrane"); in the compartments defined by these membranes (vesicles), there are placed multisets of objects (interpret them as chemical compounds), and evolution rules governing the modification of these objects in time; the objects can also pass through membranes (they can also leave the system, through the external membrane); by a maximally parallel use of these rules (in each time unit, all objects in a compartment which can evolve by the rules associated with that compartment have to evolve), one defines transitions from a configuration of the system to another configuration. A sequence of transitions is called a computation, and a result is associated with any halting computation in the form of the vector of multiplicities of objects sent out of the system during the computation. (The objects can also be described by strings, and then they evolve by string processing rules, but here we do not consider this case.)

Thus, a P system is a computing device which abstracts from a *single* cell structure and functioning. However, in most cases the cells are living together, associated in tissues, organs, organisms, and in such a framework an essential feature is that of inter-cellular communication, mainly done through the protein channels established among the membranes of the neighboring cells [20]. Because this is the main motivation for our models, we enter here into some details.

One knows (see, e.g., [1]) that the cell membrane is a bilayer of phospholipidic molecules, composed of a hydrophilic head placed toward the inner and the external spaces, and a hydrophobic tail, which is "hidden" at the middle of the two layers of molecules. The phospholipidic molecules can move with respect to each others, but preserving their orientation (always the hydrophobic tails remain "hidden") and remaining in the same plane. Because of this movement, the membrane model currently accepted is known under the name of the *fluid-mosaic model*. Very important for the chemical reactions taking place in the compartments of a cell and for the transfer of chemical compounds through membranes is the fact that in between the phospholipidic molecules there are inserted proteins. Some of them are placed in only one side of the membrane (catalytic proteins), others pass from a side of the membrane to the other side, thus establishing so-called *protein channels* between the two compartments delimited by the membrane, the inside of the cell and its environment.

Because of the fluid nature of the membrane, the protein channels can change their places on the membrane. Thus, when two cells come in contact, it may happen that protein channels from the adjacent membranes get together, establishing a common channel, which makes possible the transfer of chemical compounds from a cell to another one. After having some channels of this type, further channels are enhanced, as the two membranes remain adjacent, bound by the proteins already in contact. In this way, a complex net of channels can be established, providing a way for performing an inter-cellular communication. Fig. 1 illustrates this idea.

The suggestion to compute in such a compartmental structure was already followed in the membrane computing area, by considering P systems working on arbitrary planar graphs, see [26] (note that in the case of usual membrane structures, with 3D vesicles, the underlying graph is a tree). Such systems with a very simple graph (star, line, cycle) were proved in [26] to be computationally universal.

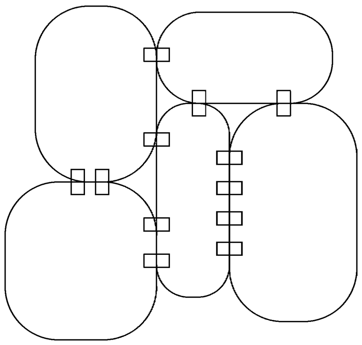

Fig. 1. Inter-cellular communication.

However, the tP systems we introduce here are different in many respects from the P systems considered in [26], in general, from usual P systems: in usual P systems one allows the evolution of objects by rules of the form $u \rightarrow v$, without any state associated with the compartments, while the rules are used in the maximally parallel mode (this corresponds to the case where each cell has only one state, and the rules are used in the mode *max*); moreover, the produced symbols have associated target indications, telling where they should be placed after using a rule. Very important in the case of P systems are various ways to control the use of rules and, mainly, to control the communication: by controlling the permeability of membranes, by associating with them electrical charges, by means of priorities among rules, etc. No such an ingredient is present in a tP system.

3. The neuron and its functioning

Because this is an important alternative motivation of our study, we present here the structure and the functioning of a neuron, basically following computer science sources (in particular [2]).

As any cell, a neuron has a *body* (also called *soma*), containing a nucleus (not of interest for what follows), and a membrane. In the case of neurons, the membrane is prolonged by two important classes of fibers, the *dendrites*, which form a very fine filamentary bush around the body of the neuron, and the *axon*, a unique, long filament, which in its turn also ends with a fine filamentous bush; each of the filaments from the end of the axon is terminated with a small bulb.

It is by means of these endbulbs and the dendrites that the neurons are linked to each other: the (electrical and chemical) impulses are sent through the axon, from the body of the neuron to the endbulbs, and the endbulbs transmit the impulses to the neurons whose dendrites they touch. Such a contact between an endbulb of an axon and the dendrites of another neuron is called a *synapse*. A neuron can be linked in this way to several other neurons, hence each neuron can receive impulses from several neurons and can transmit impulses to several neurons.

The transmission of impulses (again: electrical and chemical) from a neuron to another one is done, roughly speaking, in the following way. A neuron will be "fired" only if it gets sufficient excitation through its dendrites. These excitations should come more or less together, in a short period of time, called the *period of latent summation*. Moreover, the input impulses can be of two types, *excitatory* and *inhibitory*, and in order to get the neuron excited it is necessary that the "algebraic sum" of impulses exceed a given *threshold* specific to the neuron.

After firing a neuron and having it excited, there is a small interval of time necessary to synthesize the impulse to be transmitted to the neighboring neurons through the axon; also, there is a small interval of time necessary to the impulse to reach the endbulbs of the axon. After having an impulse transmitted through the axon, there is a time called the *refractory period* during which the axon cannot transmit another impulse.

In short, we have inputs of various types (chemical or electrical, also different in their intensities, maybe also different qualitatively, depending on the source—think to the fact that some neurons can have inputs from the various sensory areas, the skin, the eyes, the muscles), in packages (each neuron has several dendrites and they can get impulses from several other neurons or from several sensory areas), and the neuron synthesizes an impulse which is transmitted to the neurons to which it is related by synapses; the synthesis of an impulse and its transmission to the adjacent neurons are done according to certain "states" of the neuron. In different moments, depending on its previous actions, the neuron can act differently.

We will make an essential use of these observations in Section 5, when defining a formal counterpart of the biological neural nets, modeling at the same time the case of inter-cellular communication in tissue-like structures.

4. Some mathematical prerequisites

The computability notions we use here are standard and can be found in many books, for instance, in [24] and [28], so we specify only some of them, mainly in order to fix our notations.

The set of natural numbers is denoted by $\mathbf{N}$.

A *multiset* over a set X is a mapping $M : X \to \mathbf{N}$; for $a \in X$, we say that $M(a)$ is the *multiplicity* of a in M. The set $supp(M) = \{a \in X \mid M(a) > 0\}$ is called the *support* of M. Here we work only with multisets over finite sets X (implicitly, with multisets of finite support). In this case, $weight(M) = \sum_{a \in X} M(a)$ is called the *weight* of M. For two multisets M_1, M_2 over some set X we write $M_1 \subseteq M_2$ if and only if $M_1(a) \leqslant M_2(a)$ for all $a \in X$ (we say that M_1 is *included* in M_2). The *union* of M_1, M_2 is the multiset $M_1 \cup M_2 : X \to \mathbf{N}$ defined by $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$, for all $a \in X$. If $M_1 \subseteq M_2$, then we also define the *difference* multiset $M_2 - M_1 : X \to \mathbf{N}$ by $(M_2 - M_1)(a) = M_2(a) - M_1(a)$, for all $a \in X$.

For $Y \subseteq X$ and M a multiset over X, we define the *projection* of M on Y by

$$pr_Y(M)(a) = \begin{cases} M(a), & \text{if } a \in Y, \\ 0, & \text{otherwise,} \end{cases}$$

for each $a \in X$.

For a given alphabet V, we denote by V^* the language of all strings over V, including the empty string, denoted by λ. The length of $x \in V^*$ is denoted by $|x|$, while $|x|_a$ is the number of occurrences of the symbol $a \in X$ in the string $x \in V^*$. If $V = \{a_1, \ldots, a_n\}$ (the ordering of symbols is important), then $\Psi_V : V^* \to \mathbf{N}^n$, defined by $\Psi_V(x) = (|x|_{a_1}, \ldots, |x|_{a_n})$, for $x \in V^*$, is the *Parikh mapping* associated with V. The mapping Ψ_V is extended in the natural way to languages over V. For a family FA of languages, we denote by $PsFA$ the family of Parikh images of languages in FA.

By CF, CS, RE we denote the families of context-free, context-sensitive, and recursively enumerable languages, respectively. (Consequently, $PsRE$ is the family of Turing computable sets of vectors of natural numbers.)

A multiset M over an alphabet V can be represented by a string $w \in V^*$ such that $\Psi_V(w)$ gives the multiplicities in M of the symbols from V; obviously, all permutations of w are representations of the same multiset. This suggests to use strings as representations of multisets, thus, when we will say "the multiset $w \in V^*$" this means "the multiset represented by w". In this framework, we will also write $w(a)$ for denoting $|w|_a$. Clearly, the empty multiset, that with an empty support, is represented by the empty string, λ. The string representation of multisets also makes possible the writing w^n for representing the union of n copies of the multiset represented by w.

Further, more technical, notions and notations will be introduced when necessary.

5. Tissue P systems

We now pass to the definition of our variant of P systems. We introduce it in the general form, then we will consider variants of a restricted type.

A *tissue P system*, in short, a tP system, of *degree* $m \geqslant 1$, is a construct

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_{out}),$$

where:

(1) O is a finite non-empty alphabet (of *objects*);
(2) $syn \subseteq \{1, 2, \ldots, m\} \times \{1, 2, \ldots, m\}$ (*synapses* among cells);
(3) $i_{out} \in \{1, 2, \ldots, m\}$ indicates the *output cell*;
(4) $\sigma_1, \ldots, \sigma_m$ are *cells*, of the form

$$\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i), \quad 1 \leqslant i \leqslant m,$$

where:

(a) Q_i is a finite set (of *states*);
(b) $s_{i,0} \in Q_i$ is the *initial state*;
(c) $w_{i,0} \in O^*$ is the *initial multiset* of objects;
(d) P_i is a finite set of *rules* of the form $sw \to s'xy_{go}z_{out}$, where $s, s' \in Q_i$, $w, x \in O^*$, $y_{go} \in (O \times \{go\})^*$ and $z_{out} \in (O \times \{out\})^*$, with the restriction that $z_{out} = \lambda$ for all $i \in \{1, 2, \ldots, m\}$ different from i_{out}.

A tP system as above is said to be *cooperative* if it contains at least a rule $sw \to s'w'$ such that $|w| > 1$, and *non-cooperative* in the opposite case. The objects which appear in the left hand multiset w of a rule $sw \to s'w'$ are sometimes called *impulses*, while those from w' are also called *excitations*.

Remark 1. Note that we allow rules of the forms $s \to s', s \to s'w'$, although they may be considered as "non-realistic" from the biological point of view, and we also allow synapses of the form $(i,i), 1 \leqslant i \leqslant m$, although in the proofs below we do not need such circular synapses.

Any m-tuple of the form $(s_1w_1, \ldots, s_mw_m)$, with $s_i \in Q_i$ and $w_i \in O^*$, for all $1 \leqslant i \leqslant m$, is called a *configuration* of Π; thus, $(s_{1,0}w_{1,0}, \ldots, s_{m,0}w_{m,0})$ is the *initial configuration* of Π.

Using the rules from the sets $P_i, 1 \leqslant i \leqslant m$, we can define *transitions* among the configurations of the system. To this aim, we first consider three *modes of processing the impulse-objects* and three *modes of transmitting excitation-objects* from a cell to another one.

Let us denote $O_{go} = \{(a, go) \mid a \in O\}$, $O_{out} = \{(a, out) \mid a \in O\}$, and $O_{tot} = O \cup O_{go} \cup O_{out}$.

For $s, s' \in Q_i, x \in O^*, y \in O^*_{tot}$, we write

$$\begin{aligned}
sx \Rightarrow_{min} s'y \text{ iff } & sw \to s'w' \in P_i, w \subseteq x, \text{ and } y = (x - w) \cup w', \\
sx \Rightarrow_{par} s'y \text{ iff } & sw \to s'w' \in P_i, w^k \subseteq x, w^{k+1} \not\subseteq x, \\
& \text{for some } k \geqslant 1, \text{ and } y = (x - w^k) \cup w'^k, \\
sx \Rightarrow_{max} s'y \text{ iff } & sw_1 \to s'w'_1, \ldots, sw_k \to s'w'_k \in P_i, k \geqslant 1, \text{ such that} \\
& w_1 \ldots w_k \subseteq x, y = (x - w_1 \ldots w_k) \cup w'_1 \ldots w'_k, \\
& \text{and there is no } sw \to s'w' \in P_i \text{ such that} \\
& w_1 \ldots w_k w \subseteq x.
\end{aligned}$$

In the first case, only one occurrence of the multiset from the left hand side of a rule is processed (replaced by the multiset from the right hand of the rule, at the same time changing the state of the cell); in the second case a maximal change is performed with respect to a chosen rule, in the sense that as many as possible copies of the multiset from the left hand side of the rule are replaced by the corresponding number of copies of the multiset from the right hand side; in the third case a maximal change is performed with respect to all rules which use the current state of the cell and introduce the same new state after processing the objects.

We also write $sx \Rightarrow_\alpha sx$, for $s \in Q_i, x \in O^*$, and $\alpha \in \{min, par, max\}$, if there is no rule $sw \to s'w'$ in P_i such that $w \subseteq x$. This encodes the case when a cell cannot process the current objects in a given state (it can be "unblocked" after receiving new impulses from its ancestors).

From the biological point of view, the maximal mode seems more realistic, as it seems that the cells (neurons included) can process chemical objects (impulses) in a rather efficient—hence maximal—manner. Mathematically, the *max* mode corresponds to the assumption that each set P_i is union-closed, in the following sense: if two rules $sw_1 \to s'w'_1$, $sw_2 \to s'w'_2$ are in P_i, then also the rule $s(w_1 \cup w_2) \to s'(w'_1 \cup w'_2)$ is in P_i. This implies that, if P_i contains at least one rule $sw \to s'w'$ with a non-empty $w \cup w'$, then P_i is infinite (but "finitely-generated" by the union-closure of a finite set of rules).

Now, remember that the multiset w' from a rule $sw \to s'w'$ contains symbols from O, but also symbols of the form (a, go) (or, in the case of the cell i_{out}, of the form (a, out)). Such symbols will be sent to the cells related by synapses to the cell σ_i where the rule $sw \to s'w'$ is applied, according to the following three modes:

- *repl*: each symbol a, for (a, go) appearing in w', is sent to each of the cells σ_j such that $(i, j) \in syn$;
- *one*: all symbols a appearing in w' in the form (a, go) are sent to one of the cells σ_j such that $(i, j) \in syn$, non-deterministically chosen; more exactly, in the case of modes *par* and *max* of using the rules, we first perform all applications of rules, and after that we send all obtained symbols to a unique descendant of the cell (that is, we do not treat separately the objects introduced by each rule, but all of them in a package);
- *spread*: the symbols a appearing in w' in the form (a, go) are non-deterministically distributed among the cells σ_j such that $(i, j) \in syn$.

In the case of modes *repl* and *spread* there is no difference between treating the objects produced by rules in a package, as in the case of the mode *one*, or separately for each rule: for *repl* anyway the symbols are sent to all descendants, while for *spread* the distribution is completely random.

In order to formally define the transition among the configurations of Π we need some further notations. For a multiset w over O_{tot}, we denote by $go(w)$ the submultiset of symbols $a \in O$, appearing in w in the form (a, go), and by $out(w)$ the submultiset of symbols $a \in O$ appearing in w in the form (a, out). Clearly, $go(w)(a) = w((a, go))$ and $out(w)(a) = w((a, out)), a \in O$.

Moreover, for a node i in the graph defined by syn we denote $anc(i) = \{j \mid (j, i) \in syn\}$ and $succ(i) = \{j \mid (i, j) \in syn\}$ (the ancestors and the successors of node i, respectively).

Now, for two configurations $C_1 = (s_1w_1, \ldots, s_mw_m), C_2 = (s'_1w''_1, \ldots, s'_mw''_m)$ we write $C_1 \Rightarrow_{\alpha,\beta} C_2$, for $\alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$, if there are $w'_1, \ldots, w'_m$ in O^*_{tot} such that

$$s_iw_i \Rightarrow_\alpha s'_iw'_i, \quad 1 \leqslant i \leqslant m,$$

and

- for $\beta = repl$ we have:

$$w''_i = pr_O(w'_i) \cup \bigcup_{j \in anc(i)} go(w'_j);$$

- for $\beta = one$ we have:

$$w''_i = pr_O(w'_i) \cup \bigcup_{j \in I_i} go(w'_j),$$

where I_i is a subset of $anc(i)$ such that the set $anc(i)$ was partitioned into $I_1, \ldots, I_m$; at this transition, all non-empty sets of objects of the form $\bigcup_{j \in I_k} go(w'_j)$, $1 \leqslant k \leqslant m$, should be sent to receiving cells (added to multisets $w''_l, 1 \leqslant l \leqslant m$);

- for $\beta = spread$ we have:

$$w''_i = pr_O(w'_i) \cup z_i,$$

where z_i is a submultiset of the multiset $\bigcup_{j \in anc(i)} go(w'_j)$ such that $z_1, \ldots, z_m$ are multisets with the property $\bigcup_{j=1}^{m} z_j = \bigcup_{j \in anc(i)} go(w'_j)$, and such that all $z_1, \ldots, z_m$ are sent to receiving cells (added to multisets $w''_l, 1 \leqslant l \leqslant m$).

Note that in the case of the cell $\sigma_{i_{out}}$ we also remove from $w'_{i_{out}}$ all symbols $a \in O$ which appear in $w'_{i_{out}}$ in the form (a, out).

During any transition, some cells can do nothing: if no rule is applicable to the available multiset of objects in the current state, then a cell waits until new objects are sent to it from its ancestor cells. It is also worth noting that each transition lasts one time unit, and that the work of the net is synchronized, the same clock marks the time for all cells.

A sequence of transitions among configurations of the tP system Π is called a *computation* of Π. A computation which ends in a configuration where no rule in no cell can be used, is called a *halting* computation. Assume that during a halting computation the tP system Π sends out, through the cell $\sigma_{i_{out}}$, the multiset z. We say that the vector $\Psi_O(z)$, representing the multiplicities of objects from z, is *computed* by Π. (Because of the similarity with the generative work of grammars, we also use to say that the vector $\Psi_O(z)$ is *generated* by Π.)

We denote by $N_{\alpha,\beta}(\Pi), \alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$, the set of all vectors of natural numbers generated by a tP system Π, in the mode (α, β). The family of all sets $N_{\alpha,\beta}(\Pi)$, generated by all cooperative tP systems with at most $m \geqslant 1$ cells, each of them using at most $r \geqslant 1$ states, is denoted by $NtP_{m,r}(Coo, \alpha, \beta)$; when non-cooperative tP systems are used, we write $NtP_{m,r}(nCoo, \alpha, \beta)$ for the corresponding family of vector sets. When one (or both) of the parameters m, r is (are) not bounded, then we replace it (them) with $*$, thus obtaining families of the form $NtP_{m,*}(\gamma, \alpha, \beta), NtP_{*,r}(\gamma, \alpha, \beta)$, etc.

We have 18 families of the form $NtP_{*,*}(\gamma, \alpha, \beta)$, but, as we will see below, not all of them are different.

6. Two examples

Before investigating the power and the properties of tP systems, let us examine two examples, in order to clarify the definitions and to illustrate the way of working of our systems.

Consider first a simple tP system:

$$\begin{aligned}
&\Pi_1 = (O, \sigma_1, \sigma_2, \sigma_3, syn, i_{out}),\\
&\quad O = \{a\},\\
&\quad \sigma_1 = (\{s\}, s, a, \{sa \to s(a, go),\ sa \to s(a, out)\}),\\
&\quad \sigma_2 = (\{s\}, s, \lambda, \{sa \to s(a, go)\}),\\
&\quad \sigma_3 = (\{s\}, s, \lambda, \{sa \to s(a, go)\}),\\
&\quad syn = \{(1,2), (1,3), (2,1), (3,1)\},\\
&\quad i_{out} = 1.
\end{aligned}$$

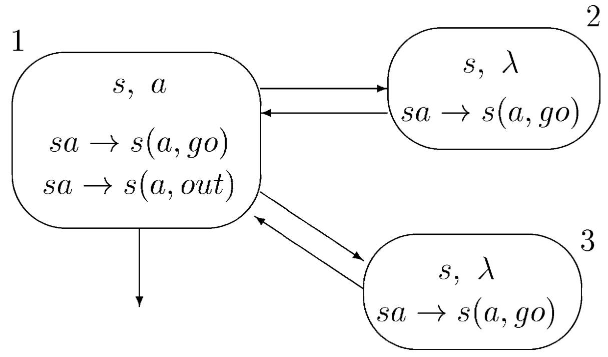

Fig. 2. An example of a tP system.

A tP system can be graphically represented as done in Fig. 2 with Π_1, with ovals associated with the cells (these ovals will contain the initial state, the initial multiset, and the set of rules, and will be labeled with $1,2,\dots,m$), with arrows indicating the synapses, and with an arrow leaving from the output cell.

The reader can easily check that we have:

$$\begin{aligned}
N_{min,repl}(\Pi_1) &= \{(n) \mid n \geqslant 1\},\\
N_{min,\beta}(\Pi_1) &= \{(1)\}, \text{ for } \beta \in \{one, spread\},\\
N_{par,repl}(\Pi_1) &= \{(2^n) \mid n \geqslant 0\},\\
N_{par,\beta}(\Pi_1) &= \{(1)\}, \text{ for } \beta \in \{one, spread\},\\
N_{max,repl}(\Pi_1) &= \{(n) \mid n \geqslant 1\},\\
N_{max,\beta}(\Pi_1) &= \{(1)\}, \text{ for } \beta \in \{one, spread\}.
\end{aligned}$$

Indeed, in the non-replicative mode of communication, no further symbol is produced, hence we only generate the vector (1). In the replicative case, the symbols produced by the rule $sa \to s(a,go)$ from cell 1 are doubled by communication. When the rules are used in the parallel mode, then all symbols are processed at the same time by the same rule, which means that all symbols present in the system are doubled from a step to the next one, therefore, the powers of 2 are obtained. When the rules are used in the minimal mode, the symbols are processed or sent out one by one, hence all natural numbers can be obtained. In the maximal mode, we can send copies of a at the same time to cells 2 and 3, and outside the system, hence again any number of symbols can be sent out.

Let us now consider a tP system with a more sophisticated functioning:

$$\Pi_2 = (\{a,a',b,b',c,c',Z\},\sigma_1,\sigma_2,\{(1,2),(2,1)\},1),$$

$$\begin{aligned}
\sigma_1 = (&\{s,s',s'',s''',s^{iv},s^{v}\},s,a,\\
&\{sa \to saa(b,go),
\end{aligned}$$

$$
\begin{aligned}
&sa \to s'aa(b,go)(b,go)(c,go),\\
&s'b \to s'',\\
&s''a \to s''a'(a,out),\\
&s''a \to s'''a'(a,out)(c',go),\\
&s'' \to s'',\\
&s'''a \to sZ,\\
&s'''b' \to s^{iv},\\
&s^{iv}a' \to s^{iv}a(a,out),\\
&s^{iv}a' \to s^{v}a(a,out)(c,go),\\
&s^{iv} \to s^{iv},\\
&s^{v}a' \to sZ,\\
&s^{v}b \to s'',\\
&sZ \to sZ\}),
\end{aligned}
$$

$$
\begin{aligned}
\sigma_2 = (&\{s,s',s''\}, s, \lambda,\\
&\{sc \to s',\\
&s'b \to s(b,go),\\
&sc' \to s'',\\
&s''b \to s(b',go)\}).
\end{aligned}
$$

If the rules are used in the parallel mode, then in the first cell we obtain 2^n copies of a, for some $n \geqslant 1$, in parallel with sending to cell 2 a number of copies of b. After changing the state of σ_1 to s' and sending c to σ_2, cell σ_1 waits two steps, while σ_2 changes its state from s to s' and then sends back to σ_1 all copies of b. At the next step, in the first cell s' is changed to s'' (and all copies of b are removed), which at the next step sends out of the net copies of each a. If this is done by using the rule $s''a \to s''a'(a,out)$, then the work of the net is never finished, because of the rule $s'' \to s''$. If we use the rule $s''a \to s'''a'(a,out)(c',go)$, then the computation stops after one more step, when c' is used in cell σ_2 (because no copy of b is present in σ_2, no further rule can be used here, hence the first cell remains "locked" in state s'''). Thus, we obtain

$$N_{\mathrm{par},\beta}(\Pi_2) = \{(2^n) \mid n \geqslant 1\}, \quad \beta \in \{repl, one, spread\}.$$

(Because each cell has only one successor, the three modes of communication are equivalent.)

Because we do not have rules involving the same pair of states (with the exception of $s''a \to s''a'(a,out)$, $s'' \to s''$, where the use of $s'' \to s''$ changes nothing, and $sa \to saa(b,go)$, $sZ \to sZ$ from cell 1, but the presence of Z means that the computation is not a successful one), the mode *max* coincides with the mode *par*, hence we get the same set of vectors as above also when using the maximal mode.

In the case of the minimal use of rules we have a more interesting behavior and we obtain:

$$N_{min,\beta}(\Pi_2) = \{(n^2) \mid n \geqslant 2\}, \quad \beta \in \{repl, one, spread\}.$$

Indeed, we start by producing a number $n \geqslant 2$ or copies of a in the first cell, while sending n copies of b to the second cell, by using the rules $sa \to saa(b, go)$ and $sa \to s'aa(b, go)(b, go)(c, go)$. During this time, the second cell waits in state s. After receiving the symbol c, the second cell changes its state to s' and then returns to s while sending a copy of b back to the first cell. During these two steps, the first cell waits in state s'.

In the presence of b, the first cell changes the state to s'', and in this state it sends out of the net one copy of a for each copy of a it has; at this time, the internal copies of a are primed. The computation can continue forever in state s'', because of the rule $s'' \to s''$, hence eventually the rule $s''a \to s'''a'(a, out)(c', go)$ must be used. If unprimed copies of a are still present, then the trap-symbol Z is introduced by the rule $s'''a \to sZ$, and the computation will never stop because of the rule $sZ \to sZ$. Thus, we have to introduce the state s''' while changing the last a into a' (and sending c' to the second cell).

In cell σ_2, the symbol c' entails the change of s into s'', which, in turn, sends to σ_1 a copy of b, primed. Now, a process as above takes place in σ_1, but controlled by states s^{iv}, s^{v} instead of s', s'': in the presence of b', the state s''' is changed to s^{iv}, which sends out a copy of a for each internal a', at the same time removing the prime; the process should stop by using the rule $s^{iv}a' \to s^{v}a(a, out)(c, go)$, otherwise the rule $s^{iv} \to s^{iv}$ can be used forever. Moreover, when using the rule which introduces the state s^{v} no primed a should be present, otherwise the trap-symbol Z is introduced by the rule $s^{v}a' \to sZ$, and the computation never stops. The state s^{iv} will return to s'' again in the presence of b.

The process is repeated as long as we have copies of b in the second cell, that is, n times. For each b one sends out of the net n copies of a, in total n^2. When no b is available, the computation stops, with the first cell in one of the states s''', s^{v} and the second cell in one of the states s', s''. Again, no difference exists between the three modes of communication.

Note the dramatic difference between the results obtained in mode *min* and in mode *par*.

7. Further language theory prerequisites

We now introduce some more technical elements from language theory, concerning matrix grammars and Lindenmayer systems.

7.1. Matrix grammars and their normal form

In the proofs of the next section we need the notion of a *matrix grammar with appearance checking*, hence we briefly introduce it here. For further details, we refer to [13].

A matrix grammar with appearance checking is a construct $G=(N,T,S,M,F)$, where N,T are disjoint alphabets, $S\in N$, M is a finite set of sequences of the form $(A_1\to x_1,\dots,A_n\to x_n)$, $n\geqslant 1$, of context-free rules over $N\cup T$ (with $A_i\in N, x_i\in(N\cup T)^*$, in all cases), and F is a set of occurrences of rules in M (N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of M are called matrices).

For $w,z\in(N\cup T)^*$ we write $w\Rightarrow z$ if there is a matrix $(A_1\to x_1,\dots,A_n\to x_n)$ in M and the strings $w_i\in(N\cup T)^*, 1\leqslant i\leqslant n+1$, such that $w=w_1, z=w_{n+1}$, and, for all $1\leqslant i\leqslant n$, either (1) $w_i=w_i'A_iw_i'', w_{i+1}=w_i'x_iw_i''$, for some $w_i',w_i''\in(N\cup T)^*$, or (2) $w_i=w_{i+1}$, A_i does not appear in w_i, and the rule $A_i\to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied—therefore we say that these rules are applied in the *appearance checking* mode.)

The language generated by G is defined by $L(G)=\{w\in T^*\mid S\Rightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When $F=\emptyset$ (hence we do not use the appearance checking feature), the generated family is denoted by MAT.

It is known that $CF\subset MAT\subset MAT_{ac}=RE$, the inclusions being proper. All one-letter languages in the family MAT are regular, see [17].

A matrix grammar $G=(N,T,S,M,F)$ is said to be in the *binary normal form* if $N=N_1\cup N_2\cup\{S,\#\}$, with these three sets mutually disjoint, and the matrices in M are in one of the following forms:
(1) $(S\to XA)$, with $X\in N_1, A\in N_2$,
(2) $(X\to Y, A\to x)$, with $X,Y\in N_1$, $A\in N_2$, $x\in(N_2\cup T)^*$,
(3) $(X\to Y, A\to\#)$, with $X,Y\in N_1$, $A\in N_2$,
(4) $(X\to\lambda, A\to x)$, with $X\in N_1$, $A\in N_2$, and $x\in T^*$.
Moreover, there is only one matrix of type 1 and F consists exactly of all rules $A\to\#$ appearing in matrices of type 3; $\#$ is called a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

According to [13], for each matrix grammar there is an equivalent matrix grammar in the binary normal form.

For an arbitrary matrix grammar $G=(N,T,S,M,F)$, let us denote by $ac(G)$ the cardinality of the set $\{A\in N\mid A\to\alpha\in F\}$. From the construction in the proof of Lemma 1.3.7 in [13] one can see that if we start from a matrix grammar G and we get the grammar G' in the binary normal form, then $ac(G')=ac(G)$.

In [15] it was proved that each recursively enumerable language can be generated by a matrix grammar G such that $ac(G)\leqslant 2$.

Consequently, to the properties of a grammar G in the binary normal form we can add the fact that $ac(G)\leqslant 2$. We will say that this is *the strong binary normal form* for matrix grammars.

7.2. Lindenmayer systems

The usual test bed for comparing the power of any new class of computing devices is the hierarchy of Turing computable functions/numbers/languages. Fine classifications of such languages (of sets of number relations, if taking the Parikh images of languages) are provided by the Chomsky hierarchy and by the Lindenmayer hierarchy. Here we consider only some basic types of L systems and we refer to [27] for details.

An *ET0L system* (extended tabled interactionless L system) is a construct $G=(V, T, w, R_1, \dots, R_n)$, where V is an alphabet, $T \subseteq V$ (the terminal alphabet), $w \in V^*$ (the axiom), and $R_1, \dots, R_n$, $n \geqslant 1$, are finite sets of context-free rules over V (called *tables*); each table R_i is *complete*, that is, for each $a \in V$ there is a rule $a \to v$ in R_i.

For $x, x' \in V^*$ and $1 \leqslant i \leqslant n$ we write $x \Rightarrow_{R_i} x'$ if $x = a_1 a_2 \dots a_k$, for $a_j \in V$, $1 \leqslant j \leqslant k$, and $x' = v_1 v_2 \dots v_k$ for $a_j \to v_j \in P_i, 1 \leqslant j \leqslant k$ (the symbols are rewritten in parallel, by the rules of the given table). The language $L(G)$ generated by G consists of all strings over T which can be derived from the axiom w by finitely many steps $\Rightarrow_{R_i}$ as above, with respect to the tables of G. The family of all such languages is denoted by $ET0L$.

If each table contains only one rule $a \to v$ for each $a \in V$, then the ET0L system is said to be *deterministic*, and the corresponding family of languages is denoted by $EDT0L$. If the system G contains only one table, then it is called an *E0L system*, and the family of the generated languages is denoted by $E0L$.

It is known that $CF \subset E0L \subset ET0L \subset CS$ and that CF (and also $E0L$) is incomparable with $EDT0L$. Moreover, $E0L$ is incomparable with MAT, and $ET0L$ contains languages which are not in MAT (but it not known whether or not $MAT - ET0L$ is non-empty).

In the sequel we will make use of the following normal form for ET0L systems. Each language $L \in ET0L$ can be generated by an ET0L system $G=(V, T, w, R_1, R_2)$, that is, with only two tables. Moreover, from the proof of Theorem V.1.3 in [27], we see that any derivation with respect to G starts by several steps of using R_1, then R_2 is used (exactly once), and the process is iterated; the derivation ends by using R_2 (actually, R_1 only prepares the string for R_2, by changing "the colors" of symbols, never changing the length of the string, while R_2 is really changing the strings).

8. The power of tP systems

As standard when considering a new computing device, we compare the power of tP systems with that of Turing machines and of restricted variants of them. We start by considering the minimal mode of using the rules, and this turns out to be computationally universal, a fact which makes natural the comparison with (Parikh images of) Chomsky families, in particular, $PsRE$. In a subsequent section we will consider the parallel and the maximal modes of using the rules, and this will make necessary the comparison with (Parikh images of) Lindenmayer families.

The following relations are direct consequences of the definitions.

Lemma 1. (i) *For all* $1 \leqslant m \leqslant m'$, $1 \leqslant r \leqslant r'$, $\gamma \in \{Coo, nCoo\}$, $\alpha \in \{min, par, max\}$, *and* $\beta \in \{repl, one, spread\}$, *we have*:

$$NtP_{m,r}(\gamma, \alpha, \beta) \subseteq NtP_{m',r'}(\gamma, \alpha, \beta) \subseteq NtP_{*,*}(\gamma, \alpha, \beta) \subseteq PsRE,$$

$$NtP_{m,r}(nCoo, \alpha, \beta) \subseteq NtP_{m,r}(Coo, \alpha, \beta).$$

(ii) *For all tP systems Π, cooperating or not, where each cell has at most one successor, and for all $\alpha \in \{min, par, max\}$ we have*

$$N_{\alpha,repl}(\Pi) = N_{\alpha,one}(\Pi) = N_{\alpha,spread}(\Pi).$$

8.1. Comparison with Chomsky families

Rather surprising, if we take into consideration the apparently weak ingredients of our models, when using the mode *min* of applying the rules, even the non-cooperative tP systems turn out to be computationally universal. (As expected, the same result holds true also when using cooperative rules, in all modes *min*, *par*, *max*.) In proving such results, we try to keep as reduced as possible both the number of cells and the maximal number of states used by the cells.

Theorem 2. $PsRE = NtP_{2,5}(\gamma, min, \beta)$, *for all* $\gamma \in \{Coo, nCoo\}$, $\beta \in \{repl, one, spread\}$.

Proof. We prove only the inclusion $\subseteq$, the opposite one can be obtained by a straightforward construction (or as a consequence of the Turing–Church thesis).

Because of the equality $PsRE = PsMAT_{ac}$, we start by considering a matrix grammar $G = (N, T, S, M, F)$ in the strong binary normal form, hence with $N = N_1 \cup N_2 \cup \{S, \#\}$, and with matrices of the four forms mentioned in Section 7.1, with only two symbols $B^{(1)}, B^{(2)} \in N_2$ used in appearance checking rules. Assume that we have k matrices of the form $m_i: (X \to \alpha, A \to x)$, with $X \in N_1, \alpha \in N_1 \cup \{\lambda\}$, $A \in N_2$, $x \in (N_2 \cup T)^*$, $1 \leqslant i \leqslant k$. Also, let lab_j, $j = 1, 2$, be such that $m_i: (X \to Y, B^{(j)} \to \#)$, $j = 1, 2, i \in lab_j$, are the matrices of M containing rules to be applied in the appearance checking manner. We assume that lab_1, lab_2, and $\{1, 2, \ldots, k\}$ are mutually disjoint.

We construct the tP system

$$\Pi = (O, \sigma_1, \sigma_2, 2),$$

with the alphabet

$$\begin{aligned} O = {} & N_1 \cup N_2 \cup T \cup \{H, Z\} \\ & \cup \{X_{i,j} \mid X \in N_1, 1 \leqslant i \leqslant k, 0 \leqslant j \leqslant k\} \\ & \cup \{A_{i,j} \mid A \in N_2, 1 \leqslant i \leqslant k, 0 \leqslant j \leqslant k\}, \end{aligned}$$

and the following cells (the axiom multiset XA of σ_1 corresponds to the initial matrix $(S \to XA)$ of the grammar G):

$$\begin{aligned} \sigma_1 = (\{ & s, s', s_f, s_1, s_2\}, s, XA, \\ & \{sX \to s'(X_{i,i}, go), \\ & s'A \to s(A_{i,i}, go) \mid m_i: (X \to \alpha, A \to x) \in M, 1 \leqslant i \leqslant k\} \\ & \cup \{sX_{i,0} \to sY \mid m_i: (X \to Y, A \to x), 1 \leqslant i \leqslant k\} \\ & \cup \{sX_{i,0} \to s_f \mid m_i: (X \to \lambda, A \to x), 1 \leqslant i \leqslant k\} \\ & \cup \{s_f D \to s_f D \mid D \in N_2 \cup \{Z\}\} \end{aligned}$$

$$\cup\{sX \to s_jY(H,go),$$
$$s_jB^{(j)} \to s_fZ,$$
$$s_jH \to s \mid j=1,2,\ m_i : (X \to Y, B^{(j)} \to \#), i \in lab_j\}$$
$$\cup\{s \to s,\ s' \to s'\}),$$

$$\sigma_2 = (\{s,s',s''\}, s, \lambda,$$
$$\{sX_{i,j} \to s'X_{i,j-1} \mid X \in N_1, 1 \leqslant i \leqslant k, 2 \leqslant j \leqslant k\}$$
$$\cup\{s'A_{i,j} \to sA_{i,j-1} \mid A \in N_2, 1 \leqslant i \leqslant k, 2 \leqslant j \leqslant k\}$$
$$\cup\{sX_{i,1} \to s''(X_{i,0},go) \mid X \in N_1, 1 \leqslant i \leqslant k\}$$
$$\cup\{s''A_{i,1} \to s(z,go)(y,out) \mid m_i : (X \to \alpha, A \to x), 1 \leqslant i \leqslant k,$$
$$x = zy, z \in N_2^*, y \in T^*\}$$
$$\cup\{s''A_{i,j} \to s''Z \mid A \in N_2, 1 \leqslant i \leqslant k, 2 \leqslant j \leqslant k\}$$
$$\cup\{s'A_{i,1} \to s''Z \mid A \in N_2, 1 \leqslant i \leqslant k\}$$
$$\cup\{s''Z \to s''Z,\ sH \to s(H,go)\});$$

moreover, we have the synapses

$$syn = \{(1,2),\ (2,1)\}.$$

(We have written (z,go) instead of $(A_1,go)\dots(A_n,go)$, with $A_i \in N_2, 1 \leqslant i \leqslant n$, such that $z = A_1 \dots A_n$, and a similar short writing was used for (y,out).)

As long as the states s,s' are present, the first cell can work at least by rules $s \to s, s' \to s'$, hence we have to remove these states. Assume that we use a rule $sX \to s'(X_{i,i},go)$ and then a rule $s'A \to s(A_{j,j},go)$, for some $1 \leqslant i,j \leqslant k$. The symbols $X_{i,i}, A_{j,j}$ are sent to the second cell, where, under the control of states s,s', one alternately decreases by one the second components of the subscripts. During this time, the first cell uses the rule $s \to s$ (no symbol from N_1 is present, so no rule of the form $sX \to s'(X_{j,j},go)$ can be used). The computation continues in this way until one of the symbols from cell 2 reaches a subscript with the second component being equal to 1. We have three cases:

Case 1: If $i<j$, then in the second cell we use the rule $sX_{i,1} \to s''(X_{i,0},go)$ and then $s''A_{j,k} \to s''Z$, for some $k \geqslant 2$. The computation will never finish, because of the rule $s''Z \to s''Z$ of the second cell.

Case 2: If $i>j$, then in the second cell we will use the rule $s'A_{j,1} \to s''Z$, which again leads to an endless computation, because the rule $s''Z \to s''Z$ can be used forever.

Case 3: If $i=j$, then after using the rule $sX_{i,1} \to s''(X_{i,0},go)$ we use the rule $s''A_{i,1} \to s(z,go)(y,out)$, which returns the second cell to state s. In cell 1 we can replace $X_{i,0}$ by Y, thus completing the simulation of the matrix $m_i : (X \to Y, A \to x)$, hence the computation can continue. In the case when m_i was a terminal matrix, then we can use the rule $sX_{i,0} \to s_f$, and then the state s_f checks whether or not any non-terminal from N_2 is still present, a case in which the computation will continue forever.

Of course, instead of using these rules we can use the rule $s \to s$, but it changes nothing, hence eventually we have to use the above mentioned rules. In this way, we can correctly simulate the matrices of G without appearance checking rules.

Assume now that we start in cell 1 by using a rule $sX \to s_jY(H, go)$, for some $m_i : (X \to Y, B^{(j)} \to \#)$ with $i \in lab_j$, $j = 1, 2$. At the next step, if the symbol $B^{(j)}$ is present, then the rule $s_jB^{(j)} \to s_fZ$ is used, which introduces the trap-symbol Z, and the computation continues forever by means of the rule $s_fZ \to s_fZ$. If no symbol $B^{(j)}$ is present, then the first cell waits one step. At this time, the second cell uses the rule $sH \to s(H, go)$ and sends back the symbol H. In the presence of this symbol, the first cell returns to state s. In this way, one correctly simulates the matrix m_i with the second rule used in the appearance checking mode.

Consequently, all derivations in G can be simulated in Π and, conversely, all halting computations in Π correspond to terminal derivations in G. One can easily see that $\Psi_T(L(G)) = N_{min,\beta}(\Pi)$, for all $\beta \in \{repl, one, spread\}$ (because no cell has two successors, the three modes of communication are equivalent). □

From the previous construction one can easily see that in the case of matrix grammars without appearance checking the states s, s', s_f of the first cell are sufficient, hence we have the following result:

Corollary 3. *$PsMAT \subseteq NtP_{2,3}(\gamma, min, \beta)$, for all $\gamma \in \{Coo, nCoo\}$, $\beta \in \{repl, one, spread\}$.*

At the price of using two more cells, we can decrease the number of used states.

Theorem 4. *$PsRE = NtP_{4,4}(\gamma, min, \beta)$, for all $\gamma \in \{Coo, nCoo\}$, $\beta \in \{one, spread\}$.*

Proof. We start again from a matrix grammar G in the strong binary normal form. With the same notations as in the proof of Theorem 2, we construct a tP system Π as indicated in Fig. 3 (we do not present this tP system formally, because the representation in the figure contains all necessary information and can also be analysed in an easier way from the point of view of the correctness of its functioning).

As usual, the matrices are of the forms $m_i : (X \to Y, A \to x)$, or $m_i : (X \to \lambda, A \to x)$, or $m_i : (X \to Y, B^{(j)} \to \#)$, Z is a trap-symbol, and H is a new symbol; D is again a generic symbol from N_2.

Let us examine the work of the tP system Π, especially because it has some features different from the tP system used in the previous proof.

In the first cell, we have to choose to pass to one of the states s' and s'' (as long as we use the rule $s \to s$ nothing is changed). In the presence of s' we send to one of cells 2 and 3 all symbols, primed. In the presence of s'' we send to one of cells 2 and 3 all symbols, double primed.

The fact that *all* symbols are communicated is ensured by the state s''', which in the presence of any symbol $D \in N_2$ introduces the trap-symbol Z, and then the rule $s'''Z \to s'''Z$ can be used forever; note that the state s''' is also introduced after simulating a terminal matrix of G, and if any $D \in N_2$ is present, then the computation will never stop.

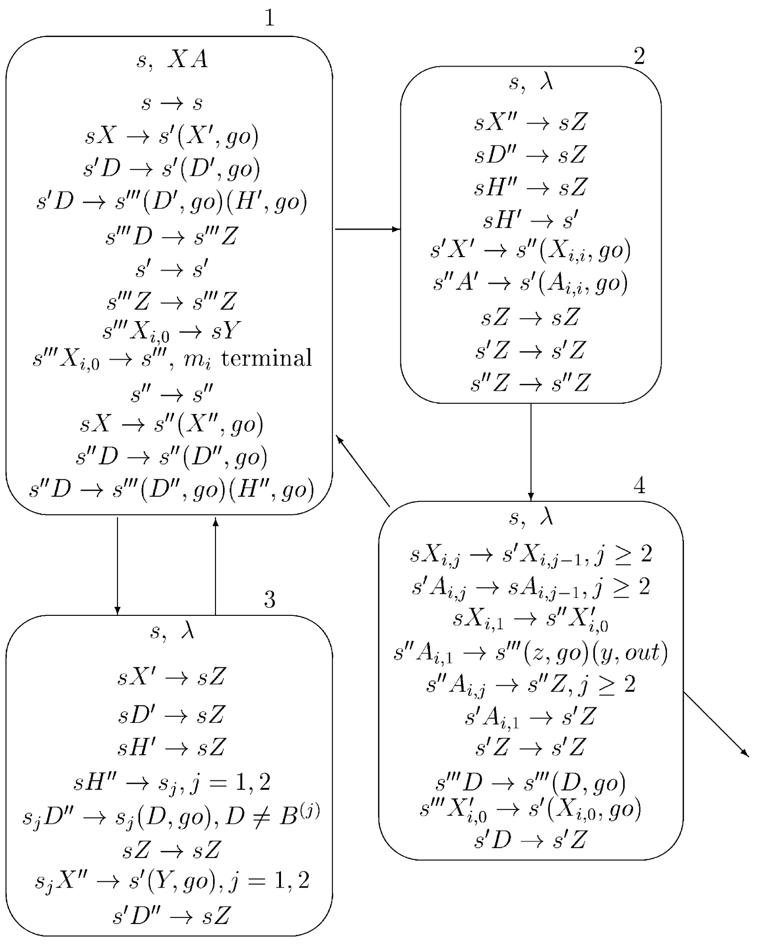

Fig. 3. The tP system from the proof of Theorem 4.

Now, if any double primed symbol arrives in the second cell, then the trap-symbol Z is introduced and the computation will never stop. Similarly, if any primed symbol arrives in the third cell, then the computation will never stop. Therefore, in the case of starting with s', all symbols arrive in cell 2. This cell, in collaboration with cell 4, simulates the matrices of G without appearance checking rules; this is performed in a way similar to the work of the tP system in the proof of Theorem 2. When a matrix $m_i, 1 \leqslant i \leqslant k$, is correctly simulated, then all symbols are sent to cell 1, by means of the state s''' (the fact that all symbols are communicated is ensured by the fact that the state s' of σ_4 will transform any remaining D from N_2 in Z and then the rule $s'Z \to s'Z$ will be used forever).

When starting in cell 1 by introducing the state s'', all symbols will be sent (double primed) to cell 3, and here we simulate a matrix $m_i : (X \to Y, B^{(j)} \to \#)$. Namely, in the presence of state $s_j, j=1,2$, all symbols D'' are sent back to cell 1, unprimed, with the exception of $B^{(j)}$.

Eventually, the rule $s_jX'' \to s'(Y, go)$ is used, completing the simulation of the matrix (and ensuring that all symbols D were sent back to cell 1 and that the symbol $B^{(j)}$ is not present: otherwise, the rule $s'D'' \to sZ$ is used, for any remaining D, including $D = B^{(j)}$, and the computation will never halt). Thus, the process is iterated.

We conclude that we have $\Psi_T(L(G)) = N_{min,\beta}(\Pi)$, for all $\beta \in \{one, spread\}$ (the fact that all primed or double primed symbols should go to the same cell ensures the fact that even in the *spread* mode we have to work in the *one* mode, otherwise the computation will never stop). □

If we use cooperative rules, then we can further decrease both the number of cells and of states. Moreover, we can characterize *PsRE* for all modes *min*, *par*, *max* of processing the objects, and this completes the study of the cooperative case.

Theorem 5. *For all* $\alpha \in \{min, par, max\}$, $\beta \in \{repl, one, spread\}$, $PsRE = NtP_{2,2}(Coo, \alpha, \beta)$.

Proof. We use again the notations from the previous two proofs, and, starting from a matrix grammar G in the strong binary normal form, we construct the tP system

$$\Pi = (N_1 \cup N_2 \cup \{X_1, X_2 \mid X \in N_1\} \cup \{c, H, Z\}, \sigma_1, \sigma_2, \{(1,2),(2,1)\}, 1),$$

with the following cells:

$$\begin{aligned}
\sigma_1 = (\{s, s'\}, s, cXA,\\
&\{scXA \to scYz(y, out) \mid (X \to Y, A \to zy) \in M, z \in N_2^*, y \in T^*\}\\
&\cup\{scXA \to sc(y, out) \mid (X \to \lambda, A \to y) \in M, y \in T^*\}\\
&\cup\{scD \to scZ \mid D \in N_2\}\\
&\cup\{scZ \to scZ\}\\
&\cup\{scX \to s'cY_j(H, go),\\
&s'cY_jB^{(j)} \to scZ,\\
&s'cY_jH \to scY \mid (X \to Y, B^{(j)} \to \#) \in M, j = 1,2\}),
\end{aligned}$$

$$\sigma_2 = (\{s\}, s, \lambda, \{sH \to s(H, go)\}).$$

With the experience of the previous proofs, the reader can easily check that we have the equality $\Psi_T(L(G)) = N_{\gamma,\beta}(\Pi)$ for all $\gamma \in \{min, par, max\}$ (all rules involve the symbol c, which appears in only one copy and is never changed, hence even in the modes *par* and *max* the rules of the first cell are actually used in the *min* mode), and for all $\beta \in \{repl, one, spread\}$ (each cell has only one successor, hence the three modes of communication coincide). □

In the case of grammars without appearance checking, the second cell is not necessary and the same assertion is true for the state s', hence we have:

Corollary 6. *$PsMAT \subseteq NtP_{1,1}(Coo, \alpha, \beta)$, for all $\alpha \in \{min, par, max\}, \beta \in \{repl, one, spread\}$.*

In the case of the parallel mode of using the rules, we can obtain sets outside *PsMAT*, even in the non-cooperative case, for all modes of communication. Indeed, for

$$\Pi = (\{a\}, (\{s\}, s, a, \{sa \to saa, sa \to s(a, out)\}), \emptyset, 1),$$

we have $N_{par,\beta}(\Pi) = \{(2^n) \mid n \geqslant 0\}$, for all $\beta \in \{repl, one, spread\}$, and this set is not in *PsMAT* [17]. The power of modes *par* and *max* will be settled in the following section.

We do not know whether or not the results in Theorems 2, 4, and 5 are optimal in the number of cells and of states. There also remains to investigate the properties of the families $NtP_{m,r}(nCoo, min, \beta)$, $\beta \in \{repl, one, spread\}$, for the small values of m, r which are not covered by the previous results.

8.2. Comparison with Lindenmayer families

The maximal mode of using the rules in a tP system resembles the parallel mode of rewriting the strings in an L system, and this makes the following results expected.

Theorem 7. *For all $\beta \in \{repl, one, spread\}$:*
(i) $PsE0L \subseteq NtP_{1,2}(nCoo, max, \beta)$,
(ii) $PsET0L \subseteq NtP_{1,3}(nCoo, max, \beta)$.

Proof. Let $G = (V, T, w, R_1, R_2)$ be an ET0L system in the normal form, hence with only two tables. We construct the tP system

$$\Pi = (V, (\{s_1, s_2, s_3\}, s_1, w, P_1), \emptyset, 1),$$

with the cell having the following rules:

$$\begin{aligned}
&s_1 a \to s_1 x,\\
&s_1 a \to s_2 x, \text{ for all } a \to x \in R_1,\\
&s_1 a \to s_3 a, \text{ for all } a \in V,\\
&s_2 a \to s_1 x, \text{ for all } a \to x \in R_2,\\
&s_3 a \to s_3 (a, out), \text{ for all } a \in T,\\
&s_3 a \to s_3 a, \text{ for all } a \in V - T.
\end{aligned}$$

In the presence of the state s_1 one simulates the table R_1; at any moment, we can switch either to state s_2, which entails the simulation of table R_2, or to state s_3, which determines the end of the computation. In the presence of s_3 no further table is simulated, but the terminal symbols are sent out of the system; if the derivation is not terminal, hence symbols from $V - T$ are still present in the cell, then the computation continues forever by using rules of the form $s_3a \to s_3a$, for $a \in V - T$, hence no output is obtained. Consequently, $\Psi_T(L(G)) = N_{max,\beta}(\Pi)$, for all $\beta \in \{repl, one, spread\}$, which proves assertion (ii).

If we have only one table, R_1, then the state s_2 is no longer necessary, hence two states suffice, and we get assertion (i). □

For tP systems working in the *min* mode, we need additional cells (and states) in order to simulate E0L and ET0L systems.

Theorem 8. *$PsE0L \subseteq NtP_{2,3}(nCoo, min, \beta)$, for all $\beta \in \{repl, one, spread\}$.*

Proof. Let $G = (V, T, w, R)$ be an E0L system. We construct the tP system

$$\Pi = (V \cup \{F, H, Z\}, \sigma_1, \sigma_2, \{(1,2),(2,1)\}, 1),$$

$$\begin{aligned}
\sigma_1 = (&\{s_1, s_2, s_3\}, s_1, w,\\
&\{s_1a \to s_1(x, go),\\
&s_1a \to s_2(x, go)(H, go) \mid a \to x \in R\}\\
&\cup\{s_2a \to s_2Z \mid a \in V\}\\
&\cup\{s_2Z \to s_2Z,\ s_1 \to s_1,\ s_2H \to s_1(F, go),\ s_2H \to s_3\}\\
&\cup\{s_3a \to s_3(a, out) \mid a \in T\}\\
&\cup\{s_3a \to s_2Z \mid a \in V - T\}),
\end{aligned}$$

$$\begin{aligned}
\sigma_2 = (&\{s, s', s''\}, s, \lambda,\\
&\{sH \to s'\}\\
&\cup\{s'a \to s'(a, go),\\
&s'a \to s''(a, go)(H, go),\\
&s''a \to sZ \mid a \in V\}\\
&\cup\{s''F \to s,\ sZ \to sZ,\ s'Z \to s'Z,\ s''Z \to s''Z\}).
\end{aligned}$$

By using rules of the form $s_1a \to s_1(x, go)$, associated with $a \to x \in R$, we simulate a derivation step in G; when all symbols are rewritten, we can use the rule $s_1a \to s_2(x, go)$ (H, go), otherwise the trap-symbol Z is introduced and the computation will never stop. If s_1 is not replaced by s_2, then the rule $s_1 \to s_1$ can be used forever. The second cell waits in state s as long as the symbol H is not received from the first cell. In the presence of H, the second cell changes its state to s', which sends back to cell σ_1 all the symbols (this is again ensured by the trap-symbol Z: if any symbol $a \in V$ is still present, then the rule $s''a \to sZ$ must be used). Now, in the presence of H, the first cell either returns from state s_2 to state s_1 (at the same time, F is sent to the second cell), and the process is iterated (when F is sent to σ_2, this cell returns to state s, waiting again for the symbol H), or to state s_3, which sends out all terminal symbols; if any symbol from $V-T$ is still present, then the computation never stops. Thus, we have the equality $\Psi_T(L(G)) = N_{max,\beta}(\Pi)$, for all $\beta \in \{repl, one, spread\}$. $\square$

In the case of ET0L systems we needed one more cell and one more state (but we do not know whether or not this result can be improved).

Theorem 9. *$PsET0L \subseteq NtP_{3,4}(nCoo, min, \beta)$, for all $\beta \in \{repl, one, spread\}$.*

Proof. Let $G = (V, T, w, R_1, R_2)$ be an ET0L system in the normal form. We construct the tP system

$$\Pi = (O, \sigma_1, \sigma_2, \sigma_3, \{(1,2),(2,1),(1,3),(3,1)\}, 3),$$

with the alphabet

$$O = V \cup \{a', a'' \mid a \in V\} \cup \{F, H, H', H'', Z\},$$

and the following cells (h', h'' are the morphisms which replace each $a \in V$ by a', a'', respectively):

$$\begin{aligned}
\sigma_1 = (&\{s_1, s_2, s_3\}, s_1, w,\\
&\{s_1 a \to s_1(h'(x), go),\\
&\ s_1 a \to s_2(h'(x), go)(H, go) \mid a \to x \in R_1\}\\
&\cup \{s_2 a \to s_2 Z \mid a \in V\}\\
&\cup \{s_2 Z \to s_2 Z,\ s_1 \to s_1\}\\
&\cup \{s_2 H \to s_1(F, go),\ s_2 H \to s_3\}\\
&\cup \{s_3 a \to s_3(h''(x), go),\\
&\ s_3 a \to s_2(h''(x), go)(H, go) \mid a \to x \in R_1\}),
\end{aligned}$$

$$\begin{aligned}
\sigma_2 = (&\{s, s', s''\}, s, \lambda,\\
&\{sH' \to s'\}\\
&\cup \{s'a' \to s'(a, go),\\
&\ s'a' \to s''(a, go)(H, go),\\
&\ s''a' \to sZ \mid a \in V\}\\
&\cup \{sZ \to sZ,\ s'Z \to s'Z,\\
&\ s''Z \to s''Z,\ s''F \to s\}\\
&\cup \{sa'' \to sZ \mid a \in V\}\\
&\cup \{sH'' \to sZ\}),
\end{aligned}$$

$$\begin{aligned}
\sigma_3 = (\{s, s', s'', s'''\}, s, \lambda,\\
&\{sH'' \to s'\}\\
&\cup\{s'a'' \to s'(x, go),\\
&s'a'' \to s''(x, go)(H, go) \mid a \to x \in R_2\}\\
&\cup\{s''a'' \to sZ \mid a \in V\}\\
&\cup\{sZ \to sZ,\ s'Z \to s'Z,\ s''Z \to s''Z,\\
&s'''Z \to s'''Z,\ s''F \to s\}\\
&\cup\{sa' \to sZ \mid a \in V\}\\
&\cup\{sH' \to sZ\}\\
&\cup\{sH'' \to s'''\}\\
&\cup\{s'''a'' \to s'''(a, out) \mid a \in T\}\\
&\cup\{s'''a'' \to sZ \mid a \in V - T\}).
\end{aligned}$$

The work of this tP system is a combination of the work of the system from the proofs of Theorems 4 and 8: all the symbols are sent either to cell σ_2, primed, or to cell σ_3, double primed, at the same time simulating the use of the table R_1 (if single primed symbols arrive in cell σ_2 or double primed symbols arrive in cell σ_1, then the computation will never stop). From σ_2 we return the symbols to the first cell, so the simulation of R_1 can be iterated. In the third cell we simulate the use of the table R_2. The trigger for starting the work of each cell (for introducing the states s' in σ_2 and σ_3, and for returning to state s_1 in σ_1) is again the symbol H (primed or double primed for σ_2 and σ_3, respectively). At any time, in cell σ_3 we can pass to the state s''', which determines the end of the simulation of the derivation. If all symbols are from T, then they are sent out of the cell and the computation stops, otherwise the computation continues forever and no output is obtained. Thus, we obtain the equality $\Psi_T(L(G)) = N_{min,\beta}(\Pi)$, for $\beta \in \{one, spread\}$.

For the case of $\beta = repl$ we remove the rules $sa'' \to sZ,\ a \in V$, and $sH'' \to sZ$ from σ_2, and the rules $sa' \to sZ, a \in V$, and $sH' \to s$ from σ_3. For the tP system Π' obtained in this way, we have $\Psi_T(L(G)) = N_{min,repl}(\Pi')$ (the replication has no effect, as all single primed symbols from σ_3 and all double primed symbols from σ_2 are ignored). □

On the other hand, the converse of assertion (ii) from Theorem 7 is also true, even in the following more general form (and this settles the study of modes *par* and *max*: they do not lead to computational universality).

Theorem 10. *$NtP_{*,*}(nCoo, \alpha, \beta) \subseteq PsET0L$, for all $\alpha \in \{par, max\}$ and $\beta \in \{repl, one, spread\}$.*

Proof. Consider a tP system $\Pi = (O, \sigma_1, \ldots, \sigma_m, syn, i_{out})$, with $\sigma_i = (Q_i, s_{i,0}, w_{i,0}, P_i)$, $1 \leqslant i \leqslant m$.

We construct an ET0L system G with the total alphabet

$$V=\{(s,i)\,|\,s\in Q_i, 1\leqslant i\leqslant m\}\cup\{(a,i)\,|\,a\in E, 1\leqslant i\leqslant m\}\cup\{Z\}\cup O$$

(Z is a new symbol), and the terminal alphabet O.

Define the morphisms $h_i:(O\cup Q_i)^*\to V^*$ by $h(\alpha)=(\alpha,i)$ for all $\alpha\in Q_i\cup O$, $1\leqslant i\leqslant m$. The axiom of G is

$$w=h_1(s_{1,0}w_{1,0})h_2(s_{2,0}w_{2,0})\dots h_m(s_{m,0}w_{m,0}).$$

The tables of G are constructed as follows.

For each $i=1,2,\dots,m$, and each $s,s'\in Q_i$, we consider the sets of rules of the following form:

$$\begin{aligned}R_i(s,s')=&\{(s,i)\to(s',i)\}\\&\cup\{(a,i)\to h_i(x)f_\beta(u)\,|\,sa\to s'xu\in P_i, x\in O^*, u\in O_{go}^*\}\\&\cup\{(q,i)\to Z\,|\,q\in Q_i-\{s\}\}\\&\cup\{(b,i)\to(b,i)\,|\text{ there is no rule } sb\to s'z \text{ in } P_i\},\end{aligned}$$

where $f_\beta(u), \beta\in\{repl, one, spread\}$, denotes a possible distribution of the symbols from u after the use of the rule $sa\to s'xu$, according to the mode β: in the case of $\beta=repl$, then

$$f_{repl}(u)=h_{j_1}(u)\dots h_{j_k}(u),$$

where $\{j_1,\dots,j_k\}=succ(i)$; in the case of $\beta=one$, then

$$f_{one}(u)=h_j(u),$$

where j is one of the descendants of cell i such that j is the same for all rules $(a,i)\to h_i(x)f_\beta(u)$ from each set of the form $R_i(s,s')$; finally, for $\beta=spread$,

$$f_{spread}(u)=h_{j_1}(u_1)\dots h_{j_k}(u_k),$$

where $\{j_1,\dots,j_k\}=succ(i)$ and $u_1,\dots,u_k$ is a partition of u. Note that for $\beta=one$ and $\beta=spread$ we have several sets of the form $R_i(s,s')$, one for each possible value of $f_\beta(u)$,

It is clear that such a set of rules, applied in the ET0L manner, corresponds to using the rules from P_i in the *max* manner, with at least a rule $sa\to s'xu$ effectively used.

In order to simulate the case when a cell does not work (no rule is applicable to its current multiset), we consider sets $R_i(s), s\in Q_i$, of rules of the following form

$$\begin{aligned}R_i(s)=&\{(s,i)\to(s,i)\}\\&\cup\{(q,i)\to Z\,|\,q\in Q_i-\{s\}\}\\&\cup\{(a,i)\to(a,i)\,|\,a\in O, \text{ and no rule } sa\to s'z \text{ appears in } P_i\}\\&\cup\{(a,i)\to Z\,|\text{ a rule } sa\to s'z \text{ there is in } P_i\}.\end{aligned}$$

When constructing the sets $R_i(s,s')$ as above for $i=i_{out}$, besides symbols $b\in O$ and $\cdot(b,go)\in O_{go}$ in the right hand side of rules $sa\to s'xu$ which lead to $(a,i)\to h_i(x)f_\beta(u)$ we can also have symbols $(b,out), b\in O$. All such symbols are replaced by b (hence they are terminal symbols with respect to G); therefore, in this case the rules of $R_i(s,s')$ are of the form $(a,i)\to h_i(x)f_\beta(u)z$, for $z\in O^*$.

Now, we perform the unions of sets $R_i(s,s'), R_i(s'')$ as above, taking exactly one set for each $i=1,2,\ldots,m$, in all possible ways; to each of the resulting sets of rules we also add completion rules of the form $c\to c$, for all symbols $c\in O\cup\{Z\}$. The obtained sets of rules are tables of G. From the way they are constructed, and from the manner of considering pair-symbols (a,i) with $a\in O$ and i specifying the cell where a is present, one can see that using such tables exactly corresponds to transitions in the tP system Π. Conversely, each transition can be simulated by using a suitable table of the ET0L system G.

What remains is to stop the work of G in the same manner as in Π, namely only for halting computations, and in that case to remove all non-terminal symbols still present in the sentential form of G. This can be done by using "terminal" tables constructed as follows. For all m-tuples of states $s_1,\ldots,s_m$ with $s_i\in Q_i, 1\leqslant i\leqslant m$, consider the set:

$$\begin{aligned}R(s_1,\ldots,s_m) = {} & \{(s_i,i)\to\lambda \mid 1\leqslant i\leqslant m\}\\ & \cup\{(a,i)\to\lambda \mid \text{no rule of the form } s_ia\to s'z \text{ is in } P_i\}\\ & \cup\{(a,i)\to Z \mid \text{there are rules } s_ia\to s'z \text{ in } P_i\}\\ & \cup\{a\to a \mid a\in O\}.\end{aligned}$$

We add all rules of the form $c\to Z$ for all symbols different from $s_1,\ldots,s_m$, a, and (a,i) for which we have already rules in the set $R(s_1,\ldots,s_m)$ (therefore, we also have the rule $Z\to Z$).

It is easy to see that such a table can be applied without introducing the trap-symbol Z, which is never eliminated from the sentential forms of G, only when no rule is applicable in Π.

Consequently, for all $\beta\in\{repl, one, spread\}$, we have the equality $\Psi_O(L(G)) = N_{max,\beta}(\Pi)$, hence the theorem is proved for the case of *max*.

The changes for the case of *par* are straightforward: in the sets $R_i(s,s')$ we have to consider only one rule of the form $(a,i)\to h_i(x)f_\beta(u)$, for a fixed $a\in O$ (that is, the set of rules is associated not only with s,s', but also with a precise rule $sa\to s'xu$ from P_i); the sets of rules $R_i(s)$ and $R(s_1,\ldots,s_m)$ remain unchanged. The parallel mode of using the rules in an ET0L system ensures the fact that when using such a rule $(a,i)\to y$ we process all occurrences of the symbol (a,i) in the current string. For the tP system Π' obtained in this way we have the equality $\Psi_O(L(G)) = N_{par,\beta}(\Pi')$, for all $\beta\in\{repl, one, spread\}$, hence the theorem is also proved for the case of the mode *par*. $\square$

Together with assertion (ii) from Theorem 7 we get the following characterization of $PsET0L$, which precisely describes the power of the mode *max* in the non-cooperative case.

Corollary 11. $NtP_{1,1}(nCoo, max, \beta) \subseteq NtP_{1,2}(nCoo, max, \beta) \subseteq NtP_{1,3}(nCoo, max, \beta) = NtP_{m,r}(nCoo, max, \beta) = NtP_{*,*}(nCoo, max, \beta) = PsET0L$, *for all* $m \geqslant 1$, $r \geqslant 3$, *and* $\beta \in \{repl, one, spread\}$.

A more precise characterization of families $NtP_{m,r}(nCoo, par, \beta)$, when $\beta \in \{repl, one, spread\}$, remains to be found (but we already know that such systems only generate Parikh images of ET0L languages).

9. Solving HPP in linear time

The architecture of tP systems and their way of working (especially the fact that in the maximal mode of using the rules we can process all objects which may be processed in such a way that the same next state is obtained, irrespective which rules are used, and the fact that in the replicative mode one can send the same objects to all successors of a cell) have an intrinsic computational power. More precisely, problems related to paths in a (directed) graph can be easily solved by a tP system, just by constructing a net with the synapses graph identical to the graph we deal with, constructing all paths in the graph with certain properties by making use of the maximal mode of applicating the rules and of the replicative communication, and checking the existence of a path with a desired property.

We illustrate this power of tP systems with the Hamiltonian Path Problem (HPP), which asks whether or not in a given directed graph $G = (V, U)$ (where $V = \{a_1, \ldots, a_m\}$ is the set of vertices, and $U \subseteq V \times V$ is the set of edges) there is a path starting in some vertex a_{in}, ending in some vertex a_{out}, and visiting all vertices exactly once. For simplicity, in what follows we assume that $a_{in} = a_1$ and $a_{out} = a_m$. It is know that the HPP is an NP-complete problem, hence it is one of the problems considered as intractable for the sequential computers (for the Turing machines).

Having a graph $G = (V, U)$ as above, we construct the tP system

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, U, m),$$

with the alphabet

$$O = \{[z; k] \mid z \in V^*, 0 \leqslant |z| \leqslant m, 0 \leqslant k \leqslant m\},$$

and with the following cells:

$$\sigma_1 = (\{s\}, s, [\lambda; 0],$$
$$\{s[\lambda; 0] \to s([1; 1], go)\}),$$

$$\sigma_i = (\{s\}, s, \lambda,$$
$$\{s[z; k] \to s([zi; k+1], go) \mid z \in V^*,$$
$$1 \leqslant |z| \leqslant m-2, |z|_i = 0, 1 \leqslant k \leqslant m-2\}),$$

for each $i = 2, 3, \ldots, m-1$, and

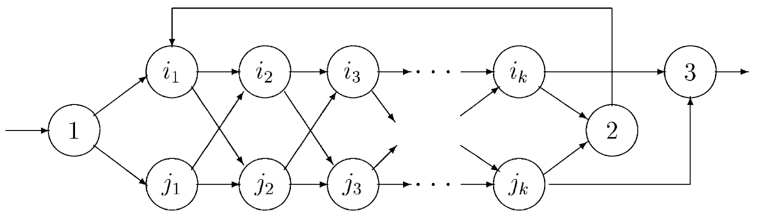

Fig. 4. A difficult graph.

$$\begin{aligned}\sigma_m = (&\{s\}, s, \lambda,\\ &\{s[z; m-1] \to s([zm; m], out) \mid z \in V^*, |z| = m-1\}).\end{aligned}$$

It is easy to see that $N_{max,repl}(\Pi) \neq \emptyset$ if and only if HPP has a solution for the graph G: the paths in G grow simultaneously in all cells of Π, because of the *max* mode of using the rules (each cell has only one state, hence all rules can be used at the same time). Moreover, the cell σ_m can work only after $m-1$ steps and a symbol is sent out of the net at the step m. Thus, it is enough to watch the tP system at step m and if any symbol is sent out, then HPP has a solution, otherwise we know that such a solution does not exist. (Note that the symbol sent out describes a Hamiltonian path in G.)

The previous construction does not work for non-cooperative tP systems with the modes *min* and *par* of using the rules. For instance, for the graph indicated in Fig. 4, one accumulates exponentially many partial paths in nodes i_r, j_r close to node 2, because the ancestors are replicating the paths, but from each cell only one path is processed. The reader is asked to simulate the work of the associated tP system for a small k, say 5.

10. Topics for further research

Compared to the wealth of research topics which can be imagined with respect to tP systems, the present paper looks rather preliminary. We list here only part of such research topics. For sure, the reader can formulate many others, looking for new variants (more adequate to the brain organization or to the tissues structure and functioning, more elegant from a mathematical point of view, more powerful, more efficient, etc.), for further problems (either of a mathematical or a computational interest, or inspired from the biology of tissues or from the theory of non-symbolic neural networks), for applications, for possible implementations. Our list roughly follows this ordering of topics, from variants to possible implementations, but it supposes no ordering according to the significance of the questions.

First, let us formulate again the question about a closer investigation of (the power of) non-cooperative tP systems working in modes *par* and *max*. Especially the latter case is important, as we consider it closer to the biochemical reality (in particular, to the brain activity), and in view of the result from Section 9, proving the computational efficiency of this mode.

Concerning the possibility of considering variants, the landscape looks endless. For instance, instead of having only one cell providing an output, we can consider several output cells; does this makes any difference? Then, between the *min* and the *par* modes of working, we can consider intermediate cases, as in the cooperating distributed grammar systems [10]: a rule should be used at most k, exactly k, at least k times in parallel, for a given k. We can also go towards *max* and use in a given step at most k, exactly k, at least k rules which may be different to each other.

Moreover, we can consider variants of the modes of communication. First, we can use no indication *go*, each new symbol being immediately sent to an adjacent cell. A more restrictive possibility is to use indications go_j, with j being the cell where the symbol must go. This last idea is related to the mode of communicating in standard P systems, from where further similar ideas can be borrowed. In the model we have considered here, the objects which are not processed at a given step are passed unchanged to the next configuration. This looks too computer science oriented; a more biologically oriented variant would be to "forget" (remove) all symbols not processed at a given step (or to remove them after a given number of steps, which could model the idea of "memory"). This seems interesting for the case of parallel and, especially, maximal use of rules, otherwise too many objects are lost. This variant is somewhat related to the case when no cell can wait, such a case leading to the end of the computation (with two possibilities: if some cells can work and some other cannot, then either the computation is blocked, without any output, or we can consider that the computation halts correctly; the first case can provide a powerful mode of controlling the computation, a sort of appearance checking feature, like in matrix grammars).

A natural variant, probably also realistic from a biological point of view, is to work in a *hybrid* mode, with various cells having various modes of using the rules and of communicating. It is also possible to control the communication by considering *filters* at the destination cells (in a similar way as in networks of language processors [9]), which is also expected to be a powerful feature.

Coming closer to language theory, a first possibility is to consider strings as results of a computation, by arranging the symbols which leave the net in the ordering of their exit; when several symbols are expelled at the same time, then any ordering of them is accepted. A completely different area of research can be obtained by using strings in the net, not multisets of symbols, that is, representing the objects from each cell as a set of strings. In such a case, the string-excitations will be handled by string processing rules, for instance, by rewriting. The presence of states provides a powerful tool for controlling the use of rules, like in state grammars [13], which will probably lead to "easy" characterizations of recursively enumerable languages (this depends on the precise definition of the mechanism).

Now, let us pass to imagining further types of problems. Consider first something which can be also considered a constructive variant: using a tP system as a transducer. Start with initial states and, maybe, objects present in cells, but also consider some inputs (dendrites linked to the environment), which bring objects into the net. After a computation, we get an output, related to this input. This is similar to the case of neural networks, from which we can borrow many other problems, for instance, related to the self-organization of the net, learning, adaptation to new environments, flexibility (adding or removing cells, states, rules, synapses). Especially this last idea, of having a tP system with a dynamic structure, seems to be promising from a computational point of view.

There are many problems of a mathematical and computational interest. In Section 8 we have found only collapsing hierarchies on the number of cells and of states. However, if we bound the number of states, for instance, at two per cell, it is highly possible to obtain an infinite hierarchy of the computed sets of vectors of natural numbers with respect to the number of cells. Various classifications of tP systems can be defined starting from the shape of the graph defined by the synapses (diameter, indegree, outdegree, planar/non-planar, ring, cycles, etc.). Which is the influence of the shape of the graph on the power and the properties of tP systems? From a computational point of view, it is of interest to consider *deterministic* cells (the solution to a problem should be obtained in a deterministic manner, a possible implementation on a usual computer needs working in a deterministic manner, etc.).

We have mentioned several times connections with other areas of distributed and parallel computing, such as classic automata networks, multiset automata, membrane computing, grammar systems and systems of cooperating automata, networks of language processors, systolic automata, and others. These connections should be more technically explored, for instance, looking for the possibility of simulating (bisimulating?) some devices by tP systems or conversely. This can bring results from an area to another one and can also suggest variants for the involved mechanisms, borrowing ideas from each other.

Going back to one of our motivations, that of having a symbolic model of the "real" neural nets, we can consider several questions of interest. What means *memory* in our context? How can we store and retrieve information in a tP system? Also, having in mind that the human brain contains a large number of competences (for processing languages, in the sense of [6], following the chomskian viewpoint that the brain competences have a grammatical structure) the question arises concerning the way of organizing a tP system-brain in such a way to handle several different classes of competences, to compute several different sets of vectors of natural numbers. What kind of "parametrization" is possible in this framework and how large effects can have such a parametrization? What about considering a universal sub-net, in the sense of programmability, embedded in a tP system, so that a large range of behaviors are obtained? Admittedly, this is a speculative question, but its interest is obvious.

What about investigating the behavior itself of tP systems, as a dynamic process, without taking (too much) care of the outputs? Cycles of configurations, deadlocks, over-crowded cells, firing of given cells, reachability of certain configurations, and so on and so forth can be notions of interest in such a case, maybe also with some biological significance.

We have reached in this way the topic of possible applications, a premature question for this stage. Further investigations are necessary before having a sensitive answer, starting, for instance, by considering several graph theory problems which can probably be approached in this framework, as already indicated in Section 9. What about problems of other types, not necessarily from graph theory? What about other modes than *max*, can they lead to polynomial (maybe, linear) solutions to NP-complete problems, as the maximal mode does?

We close this discussion by expressing our belief that the tP systems, in the form considered here or in a form which will be found to be more adequate-elegant-powerful-useful, deserve our full attention and that we hope that the domain will pay off for the research efforts which we expect to follow.

References

[1] B. Alberts, et al., Essential Cell Biology. An Introduction to the Molecular Biology of the Cell, Garland Publ. Inc, New York, London, 1998.

[2] M.A. Arbib, Brains, Machines, and Mathematics, 2nd Edition, Springer, Berlin, 1987.

[3] M.A. Arbib, The Metaphorical Brain: An Introduction to Schemes and Brain Theory, Wiley Interscience, Berlin, 1988.

[4] J.P. Banatre, P. Fradet, D. LeMetayer, Gamma and the chemical abstract reaction model: fifteen years after, in: C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View, Lecture Notes in Computer Science, Vol. 2235, Springer, Berlin, 2001, pp. 17–44.

[5] D.S. Blank, et al., Connectionist symbol processing: dead or alive? Neural Comput. Surveys 2 (1999) 1–40.

[6] C. Calude, S. Marcus, Gh. Păun, The universal grammar as a hypothetical brain, Rev. Roum. Ling. 24 (5) (1979) 479–489.

[7] C. Calude, Gh. Păun, Computing with Cells and Atoms, Taylor and Francis, London, 2000.

[8] C. Choffrut (Ed.), Automata Networks, Lecture Notes in Computer Science, Vol. 316, Springer, Berlin, 1988.

[9] E. Csuhaj-Varju, Networks of language processors, in: Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Current Trends in Computer Science, World Sci., Singapore, 2001, 781–800.

[10] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, Grammar Systems. A Grammatical Approach to Distribution and Cooperation, Gordon and Breach, London, 1994.

[11] E. Csuhaj-Varju, C. Martin-Vide, V. Mitrana, Multiset automata, in: C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing. Mathematical, Computer Science, and Molecular Computing Points of View, Lecture Notes in Computer Science, Vol. 2235, Springer, Berlin, 2001, pp. 67–82.

[12] K. Culik II, A. Salomaa, D. Wood, Systolic tree acceptors, RAIRO 18 (1984) 53–79.

[13] J. Dassow, Gh. Păun, Regulated Rewriting in Formal Language Theory, Springer, Berlin, 1989.

[14] Z. Esik, A note on isomorphic simulation of automata by networks of two-state automata, Discrete Applied Math. 30 (1991) 77–82.

[15] R. Freund, Gh. Păun, On the number of non-terminals in graph-controlled, programmed, and matrix grammars, Proc. of the MCU Conf., Chişinău, 2001, Lecture Notes in Computer Science, Vol. 2055, Springer, Berlin, 2001, 214–225.

[16] F. Gecseg, Products of Automata, Springer, Berlin, 1986.

[17] D. Hauschild, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, Acta Informatica 31 (1994) 719–728.

[18] S.C. Kleene, Representation of events in nerve nets and finite automata, Automata Studies, Princeton Univ. Press, Princeton, N.J., 1956, 2–42.

[19] M. Kudlek, C. Martin-Vide, Gh. Păun, Toward FMT (Formal Macroset Theory), Pre-proc. of Workshop on Multiset Processing, Curtea de Argeş, Romania, TR 140, CDMTCS, Univ. Auckland, 2000, pp. 149–158.

[20] W.R. Loewenstein, The Touchstone of Life. Molecular Information, Cell Communication, and the Foundations of Life, Oxford Univ. Press, New York, Oxford, 1999.

[21] A. Mateescu, V. Mitrana, Parallel finite automata systems communicating by states, Intern. J. Found. Computer Sci. 13 (2002) 733–749.

[22] W.S. McCulloch, W.H. Pitts, A logical calculus of the ideas immanent in nervous activity, Bull. Math. Biophys. 5 (1943) 115–133.

[23] M. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, Englewood Cliffs, NJ, 1967.

[24] Ch.P. Papadimitriou, Computational Complexity, Reading, MA, 1994.

[25] Gh. Păun, Computing with membranes, J. Computer Syst. Sci. 61(1) (2000) 108–143; see also Turku Center for Computer Science-TUCS Report No. 208, 1998, www.tucs.fi.

[26] Gh. Păun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, Publicationes Math. Debrecen, to appear.

[27] G. Rozenberg, A. Salomaa, The Mathematical Theory of L Systems, Academic Press, New York, 1980.

[28] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, 3 volumes, Springer, Berlin, 1997.