On languages accepted by P/T systems composed by *join*

Pierluigi Frisco
School of Mathematical and Computer Sciences, Heriot-Watt University,
EH14 4AS Edinburgh, UK
pier@macs.hw.ac.uk

Abstract

Recently the study of place/transition systems whose underlying net is composed only by *join* and *fork*, basic nets called *building blocks*, was introduced. It was proved that such study can facilitate the study and understanding of computing devices based on multiset rewriting.

Here we continue this line of research introducing *J languages* and proving that they can be accepted by place/transition systems whose underlying net is composed only by *join*. Moreover, we prove that J languages are a proper subset of context sensitive languages.

1 Introduction

The last decades have seen the birth of new fields of research at the frontier of biology, computer science, mathematics and chemistry. Two of these fields are *natural computing* and *systems biology*. *Natural computing* [17, 2] studies biological biological processes and phenomena as computational ones and it also uses such processes and phenomena to implement computation in biological media. *Systems biology* [14, 12, 4] studies the complex interactions of biological systems with the aim of an integrated understanding of them. One of the goals of systems biology is to discover and study emergent properties arising from the systemic view of biological systems. This integrative approach differs from the reductionist one used in studying biological systems.

One of the recent developments of systems biology focuses on the underlying topological interaction network of specific biological processes [1]. It has been found that such networks are mainly composed by a specific sub-networks called *building blocks* or *motifs* [13] and that the relative abundance of some of these motifs has direct relations with the emerging properties of the network.

In the present article we continue our study bridging this development of systems biology and natural computing using *Petri nets*.

Petri nets originates as mathematical representations of concurrent systems. They were introduced by C. A. Petri and since then they went through theoretical and applied developments that saw them significantly advancing the fields of parallel and distributed systems [19, 18, 5]. One of the most attractive feature of Petri nets is their simple definition capturing many of the basic notions and issues of concurrent systems. A broad range of formal systems can be classified as Petri nets [3].

In [6] a study on elementary net systems and place/transitions systems, two kinds of Petri nets, was started. There we placed, and partially answered, the

question: *What is the computational power of networks composed by specific building blocks?* The answer, continued in [8, 7], establishes some links between the topological structure of the considered systems and their computational power. Moreover, it is shown how such links can help the study of the computational power of systems based on multiset rewriting. As indicated in [6], despite our extensive research we did not find in the broad literature on Petri nets any trace of work on the same lines of what we propose.

In the present paper we continue to answer the above question introducing *J languages* and proving that they can be accepted by place/transition systems whose underlying net is composed only by *join* (a kind of building block). Moreover, we prove that J languages are a proper subset of context sensitive languages and we indicate how to relate these results to other formal systems.

2 Basic definition

We assume the reader to have familiarity with basic concepts of formal language theory [10], and in particular with the topic of place/transition systems [19, 18]. In this section we recall particular aspects relevant to our presentation.

We denote by $\mathbb{N}_1$ the set of natural numbers $\{1, 2, \ldots\}$ while $\mathbb{N} = \mathbb{N}_1 \cup \{0\}$.

Definition 1. *A* place/transition system *(*P/T system*) is a tuple* $N = (P, T, F, W, C_{in})$*, where:*

i) (P, T, F) *is a* net*:*

1. P *and* T *are sets with* $P \cap T = \emptyset$*;*
2. $F \subseteq (P \times T) \cup (T \times P)$*;*
3. *for every* $t \in T$ *there exist* $p, q \in P$ *such that* $(p, t), (t, q) \in F$*;*

ii) $W : F \to \mathbb{N}_1$ *is a* weight function*;*

iii) $K : P \to \mathbb{N}_1 \cup \{+\infty\}$ *is a* capacity function*;*

iv) $C_{in} : P \to \mathbb{N}$ *is the* initial configuration *(or* initial marking*).*

We consider P/T systems in which the weight function returns always 1 and the capacity function returns always $+\infty$. We introduced these functions for consistency with the standard definition of P/T systems. We follow the very well established notations (places are represented by empty circles, transitions by full rectangle's, tokens by bullets, etc.), concepts and terminology (configuration, input set, output set, sequential configuration graph, etc.) relative to P/T systems.

In this paper we consider P/T systems as accepting computing devices. The definition of accepting P/T systems includes the indication of a set $P_{in} \subset P$ of *input places*, one *initial place* $p_{init} \in P \setminus P_{in}$ and one *final place* $p_{fin} \in P \setminus P_{in}$. The places in $P \setminus P_{in}$ are called *work places*.

An *accepting P/T system* N with input C_{in} is denoted by $N(C_{in}) = (P, T, F, W, P_{in}, p_{init}, p_{fin})$ where $C_{in} : (P_{in} \cup \{p_{init}\}) \rightarrow \mathbb{N}$, $C_{in}(p_{init}) = 1$, is the initial configuration of the input places. So, in the initial configuration some input places can have tokens and the work place p_{init} has one token. All the remaining places are empty in the initial configuration. A configuration $C_{fin} \in \mathbb{C}_N$, the set of all reachable configurations of N, is said to be *final* (or *dead state*) if no firing is possible from C_{fin}.

We say that a P/T system $N(C_{in}) = (P, T, F, W, P_{in}, p_{init}, p_{fin})$ with $P_{in} = \{p_{in,1}, \ \ldots, \ p_{in,k}\}$, $k \in \mathbb{N}_1$, *accepts* the vector $(C_{in}(p_{in,1}), \ldots, C_{in}(p_{in,k}))$ if in the sequential configuration graph of $N(C_{in})$ there is a final configuration C_{fin} such that:

$C_{fin}(p_{fin}) > 0$;

there is at least one path from C_{in} to C_{fin};

no other configuration D in the paths from C_{in} to C_{fin} is such that $D(p_{fin}) > 0$.

The *set of vectors accepted* by N is denoted by $\mathsf{N}^k(N)$ and it is composed by the vectors $(C_{in}(p_{in,1}), \ldots, C_{in}(p_{in,k}))$ accepted by N. The just given definition of (vector) acceptance for P/T systems is new in Petri nets. Normally, the language generated by Petri nets is given by the concatenation of the labels in firing sequences. We discuss this point in Section 6.

As in [8] we call the nets *join* and *fork building blocks*, see Figure 1, where the places in each building block are distinct.

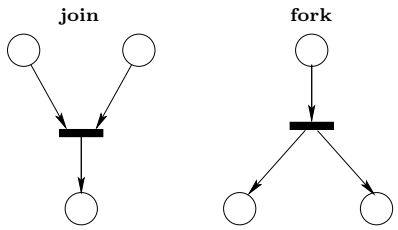

Figure 1: Building blocks: *join* and *fork*.

Also from [8] we take:

Definition 2. *Let $x, y \in \{join, fork\}$ be building blocks and let $\bar{t}_x$ and $\hat{t}_y$ be the transitions present in x and y respectively.*

We say that y comes after x *(or* x is followed by y*, or* x comes before y *or* x and y are in sequence*) if $\bar{t}_x^{\bullet} \cap {}^{\bullet}\hat{t}_y \neq \emptyset$ and ${}^{\bullet}\bar{t}_x \cap {}^{\bullet}\hat{t}_y = \emptyset$. We say that* x and y are in parallel *if ${}^{\bullet}\bar{t}_x \cap {}^{\bullet}\hat{t}_y \neq \emptyset$ and $\bar{t}_x^{\bullet} \cap {}^{\bullet}\hat{t}_y = \emptyset$.*

We say that a net is composed by *building blocks (it is* composed by x*) if it can be defined by building blocks (it is defined by* x*) sharing places but not*

transitions. So, for instance, to say that a net is composed by join *means that the only building blocks present in the net are* join*.*

In this paper we consider accepting P/T systems (in which the weight functions returns always 1 and the capacity function returns always $+\infty$) whose underlying net is composed by join. Moreover, if $N = (P, T, F, W, P_{in}, p_{init}, p_{fin})$ is such a P/T systems, then for each $t \in T$, ${}^\bullet t \in P_{in} \times P \setminus P_{in}$ and $t^\bullet \in P \setminus P_{in}$. Informally, this means that for each transition $t \in T$ the input set is given by an input place and a work place, while the output set is a work place. We call these systems *J P/T systems*.

It should be clear that J P/T systems are a normal form of accepting P/T systems: for each accepting P/T system there is a J P/T systems accepting the same language. Such J P/T systems has, eventually, more places and transitions than the original P/T system. For instance, let us assume that the net depicted in Figure 2.a is part of the net underlying an accepting P/T system N with P as set of places, $P_{in} \subset P$ as set input places and T as set of transitions. The net depicted in Figure 2.b belongs to a J P/T system N_J with $P \cup \{w'_1, w'_2\}$ as set of places, P_{in} as set of input places and $T \cup \{t'_1\}$ as set of transitions. The two nets in Figure 2 can be regarded as similar in the sets of vectors they accept.

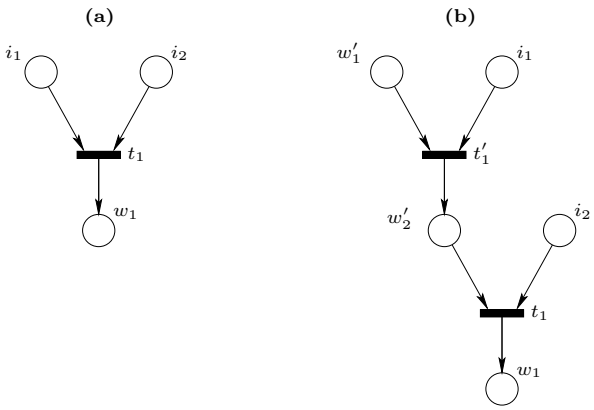

Figure 2: **(a)** a net of an accepting P/T system and **(b)** a net of a J P/T system.

3 J languages and P/T systems

In this section we prove the main result of the present paper. In order to do this, we need to introduce a new family of formal languages.

Definition 3 (J expressions, J language, length of a J expression)**.** *Let V be an alphabet, then:*

ϵ *(the* empty string*) is a J expression;*

for each $v \in V$, v *is a J expression;*

if α *and* β *are J expressions, then* $(\alpha \cup \beta)$ *is a J expression (*union*, in this case* α *and* β *are called union-terms);*

if α *and* β *are J expressions different than the empty string, then* $(\alpha\beta)$ *(*concatenation*), and* (α^+) *(*positive closure*) are J expressions;*

if $\beta_{p,j},\ 1 \leq p \leq k, 1 \leq j \leq m,\ m, k \in \mathbb{N}_1$, *are J expressions different than union and positive closure (the reason for this is explained later on), then* $\beta_{1,1}^{n_1}\ \beta_{1,2}^{n_2}\ \ldots\ \beta_{1,m}^{n_m}\ \beta_{2,1}^{n_1}\ \beta_{2,2}^{n_2}\ \ldots \beta_{2,m}^{n_m} \ldots \beta_{k,1}^{n_1}\beta_{k,2}^{n_2} \ldots \beta_{k,m}^{n_m}$ *(*exponentiation *in this case* $\beta_{p,j}$ *are called* exponentiation-terms*)* $n_j \in \mathbb{N}_1, 1 \leq j \leq m$, *are J expressions. We say that the exponentiation-termm* $\beta_{p,j}$ *are to* the power of n_j.

The language defined by a J expression α *is a* J language *and it is indicated with* $\mathsf{L}(\alpha)$.

If α *is a J expression over the alphabet* V*, then the* length *of* α *is defined as the number of symbols of* $V \cup \{\epsilon\}$ *present in* α*. The length of a J expression is indicated with* $\|\alpha\|$.

The reason why we call these languages J is because this letter is the initial one in *join*, the building block composing the nets considered in this paper.

Given a J expression α, with $\|\alpha\|_+$ we indicate the number of positive closures present in α. Let $\alpha = \delta\beta\omega$ be a J expression with δ and ω J expressions and let $\beta = \beta_{1,1}^{n_1}\beta_{1,2}^{n_2} \ldots \beta_{1,m}^{n_m}\beta_{2,1}^{n_1}\beta_{2,2}^{n_2} \ldots \beta_{2,m}^{n_m} \ldots \beta_{k,1}^{n_1}\beta_{k,2}^{n_2} \ldots \beta_{k,m}^{n_m}$, be an exponentiation. The exponentiation β is *maximal* if there are no other exponentiations in α with more than km exponentiation-terms. Given a J expression α, with $\|\alpha\|_{exp}$ we indicate the number of maximal exponentiations present in α.

In writing J expressions we can omit many parentheses is we assume that positive closure and exponentiation have precedence over concatenation or union, and that concatenation has precedence over union. If, for instance, $V = \{a, b\}$, then $\alpha = \epsilon \cup (ab^+ \cup b)^+ \cup a^n bba^n ba^n b$ is a J expression, $\|\alpha\| = 11, \|\alpha\|_+ = 2$ and $\|\alpha\|_{exp} = 1$. The elements of V present in a J expression α over V are considered numbered from left to right and are addressed with $\alpha_i, 1 \leq i \leq \|\alpha\|$. In the previous example $\alpha_1 = \epsilon$, $\alpha_2 = a$, $\alpha_3 = b$ and so on. In a similar way the positive closures and the maximal exponentiations present in a J expression are considered numbered from left to right and are addressed with $+_i,\ 1 \leq i \leq \|\alpha\|_+$ and $exp_j,\ 1 \leq j \leq \|\alpha\|_{exp}$, respectively. In the previous example $+_1 = b, +_2 = (ab^+ \cup b)$, and $exp_1 = a^n bba^n ba^n$.

The proof of the following lemma is rather long but not particularly difficult. The basic idea is to have a J P/T system in which input places are associated to the J expression defining the language accepted by the J P/T system, work places are associated with the possible union, concatenations, positive closure

and exponentiations of the J language. The J P/T system repeatedly "consumes" (accepts) one token per time from the input places and passes one token from a work place to another. The J P/T system is non-deterministic (because it "guesses" to what part of the J expression a token can be matched).

We prefer to give a unique long proof (instead of splitting it in smaller proofs) because the several parts of the proof (concatenation, related to union, positive closure and exponentiation) are interconnected.

Lemma 1. *Every J language is accepted by a J P/T system.*

Proof. Given a J language $\mathsf{L}(\alpha)$ defined by the J expression α over an alphabet V we construct a non-deterministic P/T system that can accept a vector $(C_{in}(p_{in,1}), \ldots, C_{in}(p_{in,k}))$ if and only if such vector encodes a word defined by α. Let w be a word over V, if $w \notin \mathsf{L}(\alpha)$, then the P/T system halts with no token in its final place; if $w \in \mathsf{L}(\alpha)$, then the P/T system can halt with a token in its final place.

Let $V = \{v_1, \ldots, v_n\}$ be an alphabet and $\alpha = \alpha_1 \ldots \alpha_\tau$ ($\|\alpha\| = \tau$) a J expression over V defining the J language $\mathsf{L}(\alpha)$.

The P/T system has input places named $p_{\epsilon,j}$, $p_{v_i,j}$, $1 \leq i \leq n$, $1 \leq j \leq \tau+1$. It has work places named $s_1, \ldots, s_\tau, s_{\tau+1}$ (where s_1 is the initial place and $s_{\tau+1}$ the final place), s_z, $s_1^+, \ldots, s_{\|\alpha\|_+}^+, s_1^{exp}, \ldots, s_{\|\alpha\|_{exp}}^{exp}$.

The transitions are introduced in the following:

concatenation: t_l, $1 \leq l \leq \tau$, are transitions with ${}^\bullet t_l = \{s_l, p_{\alpha_l,l}\}$ and $t_l^\bullet = \{s_{l+1}\}$ (the following items *ii* and *v* indicate exceptions to these transitions).

Informally: these transitions allow the P/T system to check if all the symbols present in α are also present in the input word. The exceptions to these check are when unions or exponentiations are present in α;

union: for every $\beta_1, \ldots, \beta_r$ J expressions such that $\beta_1 \cup \ldots \cup \beta_r$ is present in α, $\beta_1 = \beta_{1,1} \ldots \beta_{1,\|\beta_1\|} = \alpha_{q_{1,1}} \ldots \alpha_{q_{1,\|\beta_1\|}}; \ldots; \beta_r = \beta_{r,1} \ldots \beta_{r,\|\beta_r\|} = \alpha_{q_{r,1}} \ldots \alpha_{q_{r,\|\beta_r\|}}$, there are transitions:

i) $t_j^{(1)}$, $2 \leq j \leq r$, with ${}^\bullet t_j^{(1)} = \{s_{q_{1,1}}, p_{\alpha_{q_{j,1}},q_{j,1}}\}$ and:

if $|\beta_j| = 1$ and $\beta_j = \beta_j^+$, k^{th} positive closure of α ($1 \leq k \leq \|\alpha\|_+$), then ${t_j^{(1)}}^\bullet = \{s_{q_{j,1}}\}$.
Informally: if a union-term is a positive closure of only one symbol, then it is possible to check it again;

otherwise, if $|\beta_j| = 1$ and $\beta_j = \beta_j^n$ exponentiation-term of the k^{th} exponentiation of α ($1 \leq k \leq \|\alpha\|_{exp}$) to the power of n, then ${t_j^{(1)}}^\bullet = \{s_g\}$, g being the position of the first symbol of the next exponentiation-term to the power n in the same exponentiation.
Informally: if a union term is the first exponentiation-term having only one symbol, then it is possible to check the first symbol

of the next exponentiation-term to the power n of the same exponentiation;

otherwise, if $|\beta_j| = 1$, then ${t_j^{(1)}}^\bullet = \{s_{q_{r,\|\beta_r\|+1}}\}$.
Informally: if the union-term is only one symbol, then it is possible to check what after the union;

otherwise ${t_j^{(1)}}^\bullet = \{s_{q_{j,2}}\}$.
Informally: if the union-term is more than one symbol, then it is possible to check the second symbol of the same union-term;

Informally: all these transitions allow the P/T system to jump to check the remaining symbols of a union-term when the first symbol of this union-term is present;

ii) $t_j^{(2)}, 1 \leq j \leq r$, with ${}^\bullet t_j^{(2)} = \{s_{q_{j,\|\beta_j\|}}, p_{\alpha_{q_{j,\|\beta_j\|}}, q_{j,\|\beta_j\|}}\}$, ${t_j^{(2)}}^\bullet = \{s_{q_{r,\|\beta_r\|+1}}\}$ and $t_{q_{j,\|\beta_j\|}}$ from *concatenation* is not present.

Informally: when the last symbol of an element of a union-term has been checked, then the P/T system goes to check the symbol present after the last symbol of the last union-term (and not the first symbol of the next union-term). If, for instance, $\alpha = (ab \cup cd)e$ and the b was there, then the P/T system can check the e (and not the c);

positive closure: for every J expression β such that β^+ is present in α, β^+ being the $j^{th}, 1 \leq j \leq \|\alpha\|_+$, positive closure present in α and $\beta = \beta_1 \dots \beta_r = \alpha_q \dots \alpha_{q+r}$, there are transitions:

iii) $t_{q+r}^{(3)}$ with ${}^\bullet t_{q+r}^{(3)} = \{s_{q+r}, p_{\alpha_{q+r},q+r}\}$ and ${t_{q+r}^{(3)}}^\bullet = \{s_j^+\}$.
Informally: when the last symbol of β is checked the P/T system can put a token into s_j^+ the work place associated with that specific positive closure. The transitions given under *concatenation* allow the P/T system to check the symbol coming after β_r;

iv) t_j^+ with ${}^\bullet t_j^+ = \{s_j^+, p_{\alpha_q,q}\}$ and ${t_j^+}^\bullet = \{s_{q+1}\}$ if $r > 1$ or ${t_j^+}^\bullet = \{s_j^+\}$ if $r = 1$.
Informally: when a token is present in s_j^+ the P/T system checks once more the presence of the elements of the positive closure.

exponentiation: for every $\beta = \beta_{1,1}^{n_1}\beta_{1,2}^{n_2} \dots \beta_{1,m}^{n_m}\beta_{2,1}^{n_1}\beta_{2,2}^{n_2} \dots \beta_{2,m}^{n_m} \dots \beta_{k,1}^{n_1}\beta_{k,2}^{n_2} \dots \beta_{k,m}^{n_m}$ J expression in α such that: $\beta_{p,j},\ 1 \leq p \leq k,\ 1 \leq j \leq m,\ k, m \in \mathbb{N}_1$, are J expressions different than union and positive closure;
β is the $q^{th},\ 1 \leq q \leq \|\alpha\|_{exp}$, maximal exponentiation present in α,
$\beta_{p,j} = \beta_{p,j,1} \dots \beta_{p,j,\|\beta_{p,j}\|} = \alpha_{t_1} \dots \alpha_{t_{\|\beta_{p,j}\|}}$;
there are transitions:

v) $t_{q,z,j}^{(4)},\ 1 \leq z \leq k-1,\ 1 \leq j \leq m$, with ${}^\bullet t_{q,z,j}^{(4)} = \{s_{t_{x_{z,j}}}, p_{\alpha_{t_{x_{z,j}}}, t_{x_{z,j}}}\}$, ${t_{q,z,j}^{(4)}}^\bullet = \{s_{t_{y_{z,j}}}\}$, $t_{q_{x_{z,j}}}$ from *concatenation* is not present, $x_{z,j}$ is the position of the last symbol of $\beta_{z,j}$ and $y_{z,j}$ if the position of the first symbol of $\beta_{z+1,j}$.

Informally: these transitions allow the P/T system to check the presence of the first symbol of $\beta_{z+1,j}$ (and not the first symbol of $\beta_{z,j+1}$) when the last symbol of $\beta_{z,j}$ was there. If, for instance, $\alpha = (ab)^{n_1}c^{n_2}(de)^{n_1}(fg)^{n_2}$, then $\beta_{1,1} = ab, \beta_{1,2} = c, \beta_{2,1} = de, \beta_{2,2} = fg$ and during the check a b was there, then the P/T system checks the d (and not the c); if a c was there, then the P/T system checks the f (and not the d);

vi) $t^{(5)}_{q,j}$ with ${}^\bullet t^{(5)}_{q,j} = \{s_{t_x}, p_{\alpha_{t_x},t_x}\}$ and $t^{(5)\bullet}_{q,j} = \{s^{exp}_{q,j}\}$, and x is the position of the last symbol of $\beta_{p,j},\ 1 \leq p \leq k$.

Informally: when the last symbol of the last exponentiation-term to the power of n_j has been checked, then the P/T system can put a token into $s^{exp}_{q,j}$ the work place associated to the exponentiation terms with to the power of n_j;

vii) $t^{exp}_{q,j}$ with ${}^\bullet t^{exp}_{q,j} = \{s^{exp}_{q,j}, p_{\alpha_{k_1},t_1}\}$ and if $\|\beta_{p,j}\| > 1$, then $t^{exp\bullet}_{q,j} = \{s_{t_2}\}$, otherwise $t^{exp\bullet}_{q,j} = \{s_{t_y}\}$ where y is the position of the first symbol of $\beta_{1,j}$.

Informally: when a token is present in $s^{exp}_{q,j}$ the P/T system can check once more the presence of the exponentiation terms to the power of n_j. That is, it checks the presence of the first symbol of $\beta_{1,j}$;

viii) $t^{(6)}_{q,j}$ with ${}^\bullet t^{(6)}_{q,j} = {}^\bullet t^{(5)}_{q,j}$, $t^{(6)\bullet}_{q,j} = \{s_{t_y}\}$ where y is the position of the first symbol of $\beta_{1,j+1}$ (or the first symbol after the exponentiation) and t_w from *concatenation* is not there (where w is the position of the last symbol of $\beta_{k,j}$);

Informally: the cycle checking the exponentiation-terms to the power of n_j can be interrupted when the last symbol of $\beta_{k,j}$ has been checked. When this happens the first symbol of $\beta_{1,j+1}$ is checked (and not what comes after the last symbol of $\beta_{k,j}$). In the previous example if an e was there, then the P/T system can check the c (and not the f);

extra: $t_{v_i,j}$ with ${}^\bullet t_{v_i,j} = \{s_{\tau+1}, p_{v_i,j}\}, t^\bullet_{v_i,j} = \{s_z\}$ for each $1 \leq i \leq n, 1 \leq j \leq \tau + 1$.

Informally: if when $s_{\tau+1}$ (final place) has a token any of the input places contain at least one token, then the P/T system removes the token from the final place.

All these transitions belong to *join*. An example of the construction of such a P/T system is given in Section 4.1.

The P/T system is such that the firing of every transition removes one token from an input place and one token from a work place and put one token in a work place. This means that the number of tokens in the work places is constant (invariant) and equal to 1 for all the computation of the P/T system.

The initial configuration of the P/T system has always one token in s_1. Additionally, several tokens can be present in the input places.

The P/T system is such that if $C_{in}(p_{w_1,j_1}) = k_1 \dots C_{in}(p_{w_t,j_t}) = k_t,\ w_i \in V \cup \{\epsilon\},\ 1 \leq j \leq \tau,\ 1 \leq i \leq t,\ t \in \mathbb{N}_1$, then the P/T system halts with a token in $s_{\tau+1}$ if and only if $w_1^{k_1} \dots w_t^{k_t}$ belongs to $\mathsf{L}(\alpha)$.

The P/T system works in the following way: most of the firing of transitions check if the tokens present in the input places can be matched with the symbols in the J expression α defining the language $\mathsf{L}(\alpha)$. If the firing occurs, then such matching is present, if not the P/T system will never have a token in its final place $s_{\tau+1}$.

The possible checking firing sequences of the P/T system are:

concatenation: for each $\beta = \beta_1 \dots \beta_{\|\beta\|},\ \beta_1, \dots, \beta_{\|\beta\|} \in V \cup \{\epsilon\}$, J expression present in α, the firing sequence checking if $\beta_1, \dots, \beta_{\|\beta\|}$ are consecutive symbols in the input word is possible;

union: for each $\beta_1 \cup \dots \cup \beta_r$ J expression present in $\alpha,\ \beta_1, \dots, \beta_r$, being J expressions, the firing sequences checking if any of the $\beta_i, 1 \leq i \leq r$, is present in the input word is possible;

positive closure: for each β^+ J expression present in α, the firing sequence checking if any consecutive repetition of β is present in the input word is possible;

exponentiation: for each $\beta = \beta_{1,1}^{n_1}\beta_{1,2}^{n_2} \dots \beta_{1,m}^{n_m}\beta_{2,1}^{n_1}\beta_{2,2}^{n_2} \dots \beta_{2,m}^{n_m} \dots \beta_{k,1}^{n_1}\beta_{k,2}^{n_2} \dots \beta_{k,m}^{n_m}$ J expression present in α such that $\beta_{p,j},\ 1 \leq p \leq k,\ 1 \leq j \leq m,\ k, m \in \mathbb{N}_1$, are J expressions different than union and positive closure, the firing checking that all the $\beta_{p,j}$ exponentiation-terms are present n_j times is possible.

It can happen that after any of these checking firing sequences the P/T system reaches a configuration having a token in $s_{\tau+1}$. When this happens another (last) firing can take place only if any of the input places has a token.

If this happens, then the token from $s_{\tau+1}$ is removed.

It is important to notice that in any configuration of a J P/T system only one (any one) work place contains just one token.

In order to complete this proof we have to prove that if the P/T system has an initial configuration such that $C_{in}(p_{w_1,j_1}) = k_1, \dots, C_{in}(p_{w_t,j_t}) = k_t,\ w_i \in V \cup \{\epsilon\},\ 1 \leq j \leq \tau,\ 1 \leq i \leq t,\ t \in \mathbb{N}_1$ and $w_1^{k_1} \dots w_t^{k_t}$ does not belong to $\mathsf{L}(\alpha)$, then the P/T system will halt with no token in $s_{\tau+1}$.

We start noticing that if in the initial configuration of the P/T system there are tokens in $p_{v,j}$ and $p_{v',j},\ v, v' \in V \cup \{\epsilon\},\ v \neq v',\ 1 \leq j \leq \tau + 1$, then the last configuration of the P/T system sees no token in $s_{\tau+1}$. This is because, considering how the P/T system has been defined, there are not two transitions t and t' different than *extra* such that $\{p_{v,j}\} \in {}^\bullet t$ and $\{p_{v',j}\} \in {}^\bullet t'$. So, the P/T system can halt in a configuration with a token in $s_{\tau+1}$ only if in its initial configuration there are no two places $p_{v,j}$ and $p_{v',j}$ with at least one token each.

By contradiction, let us assume that the P/T system has an initial configuration such that $C_{in}(p_{w_1,j_1}) = k_1, \ldots, C_{in}(p_{w_t,j_t}) = k_t,\ w_i \in V \cup \{\epsilon\},\ 1 \leq j \leq \tau,\ 1 \leq i \leq t,\ t \in \mathbb{N}_1$, that $w_1^{k_1} \ldots w_t^{k_t}$ does not belong to $\mathsf{L}(\alpha)$ and that the P/T system halts with a token in $s_{\tau+1}$. This means that from the initial configuration there is a firing sequence (following the concatenations, unions, positive closures and exponentiations present in α) having as last configuration one in which $s_{\tau+1}$ has a token. But this implies that $w_1^{k_1} \ldots w_t^{k_t}$ belongs to α. A contradiction. □

In Section 4 we present some examples related to this lemma.

Before presenting the next results we explain why exponentiation-terms have to be different than union and positive closure. Let $\beta = \beta_1^{n_1}\beta_2^{n_2}\beta_3^{n_1}\beta_4^{n_2}$ be an exponentiation with $\beta_1^{n_1}, \beta_2^{n_2}, \beta_3^{n_1}, \beta_4^{n_2}$ exponentiation terms. There is no meaning in having (for instance) $\beta_1 = \alpha^+$, where α is a J expression, as $\beta_1^{n_1} = \alpha^{+^{n_1}} = \alpha^+$. So, $\beta = \alpha^+\beta_2^{n_2}\beta_3^{n_1}\beta_4^{n_2}$ is the concatenation of α^+ to an exponentiation.

The reason why exponentiation-terms cannot be union depends on the fact that J P/T systems do not have memory. Let β be defined as in the above, let $n_2 > 1$ and let (for example) $\beta_2 = \alpha_1 \cup \alpha_2$, where α_1 and α_2 are J expressions. This means that $\beta = \beta_1^{n_1}(\alpha_1 \cup \alpha_2)^{n_2}\beta_3^{n_1}\beta_4^{n_2}$. Let us assume that in the initial configuration of the J P/T system accepting β there are some tokens in the input places associated to α_1 and to α_2. We know from Lemma 1 that the check of the presence of symbols in β_2 and β_4 is done in passages: first checking the occurrence of symbols in β_2, then the one in β_4, then (second passage) the one in β_2 again, and so on. It can be that (as the J P/T system does not have memory) in the first passage tokens related to α_1 are checked, while in the second passage tokens related to α_2 are checked. This would not be a desired behaviour.

The fact that exponentiation-terms cannot be union is not a big limit as we can rewrite β as $\beta_1^{n_1}\alpha_1^{n_2}\beta_3^{n_1}\beta_4^{n_2} \cup \beta_1^{n_1}\alpha_2^{n_2}\beta_3^{n_1}\beta_4^{n_2}$.

Here a concept that we need in the following:

Definition 4 (cycle, lenght of a cycle)**.** *Let N be a J P/T system. We say that N contains cycles if and only if some firing sequences of N are of the kind $\alpha\beta^n\gamma \in T^*$, where T is the set of transitions of N and $n > 1$. A* cycle *is a cyclic path in the net underlying N having β as sequential transitions in a firing sequence.*

We denote cycles with the sequence of pairs of places and transitions belonging to it. The length of a cycle *is the number of transitions present into it.*

In Figure 3 a cycle of length 2 is $(s_1^+, p_{a,2})t_1^+(s_3, p_{a,3})t_3^{(3)}$.

Here the converse of the previous lemma:

Lemma 2. *J P/T systems can accept J languages.*

Proof. We only provide a sketch of the proof. It is very important to recall that:

the underlying topological structure of J P/T systems is composed by *join* and that for each transition the input set is given by an input place and a work place;

the initial configuration sees tokens in input places and in only one work place (the initial place).

Let N be a J P/T system and let its input places be associated to symbols in an alphabet V. If N contains no cycle, then N accepts concatenations of symbols and unions of symbols and their concatenation. If instead N contains cycles, then this means that concatenations of symbols can be repeatedly checked. This means that N can accept the positive closure of symbols, concatenations and their union.

Now we have to prove that N can accept exponentiations. Let us assume that N accepts $\beta_1^+\beta_2^+$ with $\beta_1 = \beta_{1,1}\beta_{1,2}\dots\beta_{1,k_1},\ \beta_2 = \beta_{2,1}\beta_{2,2}\dots\beta_{2,k_2},\ \beta_{1,i}, \beta_{2,j} \in V^+,\ 1 \leq i \leq k_1,\ 1 \leq j \leq k_2$. In order to simplify the proof we assume that $k_1 = k_2$. With slight modifications the result holds also if $k_1 \neq k_2$.

It is possible to define another J P/T system N' accepting $\beta_{1,1}^{n_1}\beta_{1,2}^{n_2}\dots\beta_{1,k_1}^{n_{k_1}}$ $\beta_{2,1}^{n_1}\beta_{2,2}^{n_2}\dots\beta_{2,k_1}^{n_{k_1}}$. The system N' is very similar to N. It is made such that when the last symbol of $\beta_{1,1}$ is checked, then the first symbols of $\beta_{2,1}$ is checked. When the last symbol of $\beta_{2,1}$ is checked, then the system can either check the first symbol of $\beta_{1,1}$ or the first symbol of $\beta_{1,2}$ and so on. Informally: for J P/T systems exponentiation is a shuffling of concatenations. □

From the previous two lemmas we have:

Theorem 1. *J P/T systems can only accept J languages.*

4 Examples related to Lemma 1

In this section we describe the J P/T system having a finite number of places and transitions and whose underlying net is composed only by *join* accepting the J language defined by the J expression $\alpha = \epsilon \cup (aa)^+ \cup a^{n_1}b^{n_2}a^{n_1}\epsilon^{n_2}$ for $n_1, n_2 \in \mathbb{N}_1$. This is a rather simple J expression, anyhow we will see that the P/T system accepting $\mathsf{L}(\alpha)$ (partially depicted in Figure 3) is complex.

4.1 Creation of the P/T system

We consider the alphabet $V = \{a, b\}$ and the J expression $\alpha = \epsilon \cup (aa)^+ \cup a^n bba^n$, so we have $\|\alpha\| = 7 = \tau,\ \|\alpha\|_+ = 1,\ \|\alpha\|_{exp} = 1$. The input places of the P/T system are: $p_{\epsilon,j},\ p_{a,j}$ and $p_{b,j},\ 1 \leq j \leq 8 = \tau + 1$; the work places are: $s_1, \dots, s_8$ (where s_1 is the initial place and s_8 is the final place), s_z, s_1^+ and s_1^{exp}.

The transitions are introduced in the following:

concatenation: t_1 is not present, $t_1^{(2)}$ is present instead;

t_2 is present with ${}^\bullet t_2 = \{s_2, p_{a,2}\}$ and $t_2^\bullet = \{s_3\}$;

t_3 is not present, $t_2^{(2)}$ is present instead;

t_4 is not present, $t_{1,1}^{(4)}$ is present instead;

t_5 is not present, $t_{1,1,2}^{(4)}$ is present instead;

t_6 is not present, $t_{1,1}^{(6)}$ is present instead;

t_7 is present with ${}^\bullet t_7 = \{s_7, p_{\epsilon,7}\},\ t_7^\bullet = \{s_8\}$;

union: $t_2^{(1)}$ is present with ${}^\bullet t_2^{(1)} = \{s_1, p_{a,2}\}$ and ${t_2^{(1)}}^\bullet = \{s_3\}$;

$t_3^{(1)}$ is present with ${}^\bullet t_3^{(1)} = \{s_1, p_{a,4}\}$ and ${t_3^{(1)}}^\bullet = \{s_7\}$;

$t_1^{(2)}$ is present with ${}^\bullet t_1^{(2)} = \{s_1, p_{\epsilon,1}\}$ and ${t_1^{(2)}}^\bullet = \{s_8\}$;

$t_2^{(2)}$ is present with ${}^\bullet t_2^{(2)} = \{s_3, p_{a,3}\}$ and ${t_2^{(2)}}^\bullet = \{s_8\}$;

$t_3^{(2)}$ is present with ${}^\bullet t_3^{(2)} = \{s_7, p_{a,7}\}$ and ${t_3^{(2)}}^\bullet = \{s_8\}$;

positive closure: $t_3^{(3)}$ is present with ${}^\bullet t_3^{(3)} = \{s_3, p_{a,3}\}$ and ${t_3^{(3)}}^\bullet = \{s_1^+\}$;

t_1^+ is present with ${}^\bullet t_1^+ = \{s_1^+, p_{a,2}\}$ and ${t_1^+}^\bullet = \{s_3\}$;

exponentiation: $t_{1,1,1}^{(4)}$ is present with ${}^\bullet t_{1,1,1}^{(4)} = \{s_4, p_{a,4}\}$ and $t_{1,1,1}^{(4)\bullet} = \{s_6\}$;

$t_{1,1,2}^{(4)}$ is present with ${}^\bullet t_{1,1,2}^{(4)} = \{s_5, p_{b,5}\}$ and $t_{1,1,2}^{(4)\bullet} = \{s_7\}$;

$t_{1,1}^{(5)}$ is present with ${}^\bullet t_{1,1}^{(5)} = \{s_6, p_{a,6}\}$ and $t_{1,1}^{(5)\bullet} = \{s_{1,1}^{exp}\}$;

$t_{1,2}^{(5)}$ is present with ${}^\bullet t_{1,2}^{(5)} = \{s_7, p_{\epsilon,7}\}$ and $t_{1,2}^{(5)\bullet} = \{s_{1,2}^{exp}\}$;

$t_{1,1}^{exp}$ is present with ${}^\bullet t_{1,1}^{exp} = \{s_{1,1}^{exp}, p_{a,4}\}$ and $t_{1,1}^{exp\bullet} = \{s_6\}$;

$t_{1,2}^{exp}$ is present with ${}^\bullet t_{1,2}^{exp} = \{s_{1,2}^{exp}, p_{b,5}\}$ and $t_{1,2}^{exp\bullet} = \{s_7\}$;

$t_{1,1}^{(6)}$ is present with ${}^\bullet t_{1,1}^{(6)} = \{s_{1,1}^{exp}, p_{a,6}\}$ and $t_{1,1}^{(6)\bullet} = \{s_5\}$.

extra: $t_{a,j}$ are present with ${}^\bullet t_{a,j} = \{s_8, p_{a,j}\}, t_{a,j}^\bullet = \{s_z\}$ and $t_{b,j}$ are present with ${}^\bullet t_{b,j} = \{s_8, p_{b,j}\},\ t_{b,j}^\bullet = \{s_z\},\ 1 \leq j \leq 8$.

The just described P/T system is partially depicted in Figure 3. All the work places and only the input places $p_{\epsilon,1}, p_{a,2}, p_{a,3}, p_{a,4}, p_{b,5}, p_{a,6}$ and $p_{\epsilon,7}$ are depicted; all transitions except $t_1^{(8)}$ and except the ones related to *extra* are depicted, too.

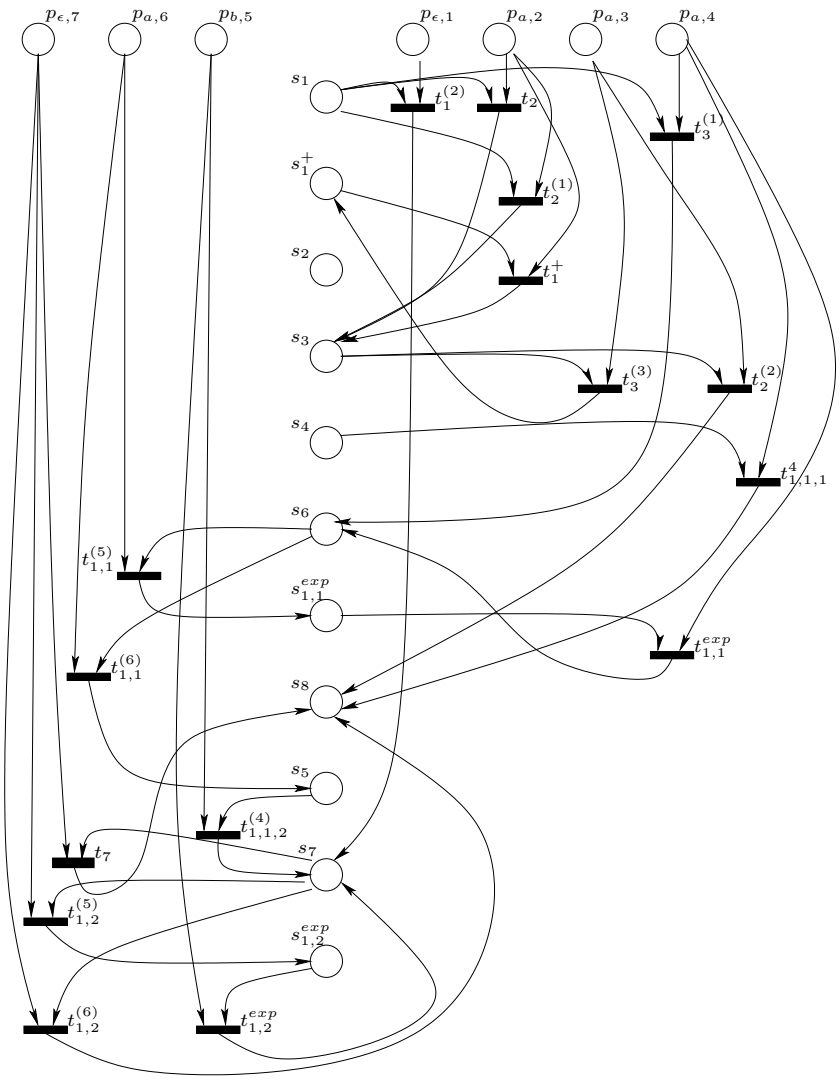

Figure 3: Partially depicted net underlying the P/T system defined in Section 4.1.

4.2 Initial configurations and firing sequences

Here we consider five initial configurations and describe some of the firing sequences for each of these configurations. We recall that, as the P/T system is non-deterministic, for each initial configuration the system can have more than one firing sequence.

(First case) There is one token in s_1, two tokens in $p_{a,4}$, one token in $p_{b,5}$, two tokens in $p_{a,6}$ and one token in $p_{\epsilon,7}$. One firing sequence associated to this initial configuration is $t_3^{(1)}t_{1,1}^{(5)}t_{1,1}^{exp}t_{1,1}^{(6)}t_{1,1,2}^{4}t_{1,2}^{(6)}$.

The firing of $t_{1,6}^{(6)}$ let a token to be put in s_8, final state of the P/T system, and no other firing occurs. In this way the word is accepted reflecting the fact that $a^2ba^2 \in \mathsf{L}(\alpha)$.

Another firing sequence associated to this initial configuration is $t_3^{(1)}t_{1,1}^{(6)}t_{1,1,2}^{(4)}t_{1,2}^{(6)}t_{a,4}$. The firing of $t_{a,4}$ let a token to be put in s_z. The word is not accepted.

(Second case) There is one token in s_1, four tokens in $p_{a,4}$ and one token in $p_{b,5}$.

The only possible firing sequence associated to this initial configuration is: $t_3^{(1)}$. The input vector is not accepted and there is no other firing sequence accepting the initial configuration. This reflects the fact that $a^4b \notin \mathsf{L}(\alpha)$.

(Third case) There is one token in s_1, one token in $p_{a,4}$, one token in $p_{b,5}$, two tokens in $p_{a,6}$ and one token in $p_{\epsilon,7}$.

The firing sequences $t_3^{(1)}t_{1,1}^{(5)}$ and $t_1^{(3)}t_{1,1}^{(6)}t_{1,1,2}^{(4)}t_{1,2}^{(6)}t_{a,6}$ are associated to this initial configuration. In both cases the input vector is not accepted and there is no other firing sequence accepting the initial configuration. This reflects the fact that $aba^2 \notin \mathsf{L}(\alpha)$.

(Fourth case) There is one token in s_1, one token in $p_{a,2}$ and one token in $p_{a,3}$. One firing sequence associated to this initial configuration letting the P/T system not to accept the initial vector is $t_2t_3^{(3)}$, while firing sequences letting the P/T system to accept the initial vector are: $t_2t_2^{(2)}$ and $t_2^{(1)}t_2^{(2)}$. This reflects the fact that $aa \in \mathsf{L}(\alpha)$.

(Fifth case) There is one token in s_1, two tokens in $p_{a,2}$ and two tokens in $p_{a,3}$. Two firing sequence associated to this initial configuration letting the P/T system not to accept the initial vector are: $t_2t_2^{(2)}t_{a,2}$ (this last transition is not depicted in Figure 3), $t_2t_3^{(3)}t_1^{+}t_3^{(3)}$, while one firing sequence letting the P/T system to accept the initial vector is: $t_2t_3^{(3)}t_1^{+}t_2^{(2)}$. This reflects the fact that $a^2a^2 \in \mathsf{L}(\alpha)$.

5 J languages and other families of languages

In this section we present a few results relating J languages to other classes of formal languages. We denote with CF and CS the classes of context-free and context sensitive language, respectively.

Lemma 3. *J languages* $\subseteq$ CS.

Proof. It is easy to show that a linear bounded Turing machine can simulate any J P/T system. This is because J P/T systems:

have a finite description;

have a finite initial configuration;

do not generate tokens during their computations (that is, tokens are only consumed).

Given a J P/T system N it is possible to define a Turing machine having the description of N in its finite control and having as input an encoding of the input of the input of N. This encoding can be, for instance, of the kind $p_1 \bullet \bullet \bullet p_2 \bullet \bullet \ldots$ where p_1 and p_2 denote input places of N and the number of $\bullet$ after such a place denotes the number of tokens present in the place in the initial configuration. To simulate that a token is removed from an input place the Turing machine changes one $\bullet$ (associated to the input place) into another symbol. The work place currently having a token (as said in the proof of Lemma 1 in any configuration of a J P/T system only one - any one - work place contains just one token) can be also recorded in the finite control of the Turing machine.

It should be clear that the Turing machine simulate N without using any extra tape. □

Lemma 4. *J languages* $\neq$ CS.

Proof. We prove that $a^{2^n},\ n \in \mathbb{N}_1$, (a context free language) is not a J language.

First of all we notice that if a J P/T system has t input places and k work places, then it will have at most tk^2 transitions. This is because the input set of each transition is composed by a input place and a working place while the output set is a working place (that could be the same as the working place in the input set). This means that if such a J P/T system has more than tk^2 tokens in its initial configuration and it accepts that configuration, then at least one accepting firing sequence associated to that initial configuration has some cycle.

By contradiction, let N be a J P/T system accepting $a^{2^n},\ n \in \mathbb{N}_1$, having tk^2 transitions. Moreover, let C_{in} be an initial configuration of N with more than tk^2 tokens such that N accepts C_{in} and let FS be an accepting firing sequence associated to that initial configuration. There is an unbounded number of initial configurations accepted by N that can be derived by C_{in}. These configurations are the ones having one token more in each of the input places present into cycles of FS. These initial configurations are accepted because N loops once

more in one of its cycles. If, for instance, $FS = \alpha\beta^n\gamma$, then also the firing sequence $\alpha\beta^{n+1}\gamma$ is an accepting firing sequence of N. But this means that if the length of the cycle if c and there is an $m \in \mathbb{N}_1$ such that a^{2^m} is accepted by N, then also $a^{2^m}a^c$ is accepted by N. A contradiction. □

From the previous two lemmas we have:

Theorem 2. *J languages* $\subset$ CS.

Clearly regular languages are a subset of J languages and for sure J languages contain some non-CF language as, for instance, $a^nbc^nde^n$ but we were not able to be more precise than this. So, we were not able to define a reasonable lower bound for J languages.

It would be very interesting to know how J languages place themselves in respect to language generated by Lindenmayer systems (as it is known that 0L $\subset$ E0L $\subset$ T0L $\subset$ ET0L $\subset$ CS and that CF $\subset$ E0L).

6 Final remarks

In the Introduction we said that the study reported in the present article bridges the network development of systems biology and natural computing. At the moment this bridging is only abstract: we relate topologies to languages. Clearly, this research falls in the realm of formal language theory. Anyhow, knowing from systems biology that some biological processes have specific underlying networks, this kind of results could be of interest when questions about reachability of certain configurations are asked for biological networks. In this case knowing that a network with certain motifs generate certain languages can be essential in answering this kind of questions.

In Section 2 we said that the way to accept languages (sets of vectors) considered by us differs from the standard one used in Petri nets (concatenations of the labels of firing sequences) [9, 11]. There are two reasons why we did not considered this standard way in the present paper: we wanted to focus only on the topology and we are in the process of writing a paper discussing the relations between these two different ways to accept languages.

In [7, 8] it is shown how the results obtained from the computational power of P/T system whose underlying net is composed by *join* and *fork* can facilitate the study of the computational power of models of *membrane systems* (also known as *P systems*) [16] based on multiset rewriting. These results use the definitions of *strong* and *weak* equivalence (also present in [7, 8]).

In a nutshell the idea is the following: if a formal system S can simulate *fork*, *join* and their composition, then the results on the computational power of P/T systems whose underlying net is composed by *join* and *fork* are also valid to S.

In [7, 8] it is shown that P systems with catalysts can simulate a *fork* using rules of the kind $a \to b_1b_2$, while the simulation of a *join* does not require the use of such rules. So, knowing from [7, 8] how P systems with catalysts can

simulate *join* and knowing Theorem 1, we can say that the family of languages generated by P systems with catalysts not using rules of the kind $a \to b_1 b_2$ is J.

Using the definitions and results of P systems with catalysts present in [7, 8] we can be more precise and state:

Corollary 1.

the family of languages accepted by P systems with catalysts of degree 2 and 2 catalysts not using rules of the kind $a \to b_1 b_2$ is J;

the family of languages accepted by purely catalytic P systems of degree 2 and 3 catalysts not using rules of the kind $a \to b_1 b_2$ is J.

In this paper we continued the study of the computational power of P/T systems in terms of *join* and *fork*, their combinations and the functions W and K.

References

[1] U. Alon. *An Introduction to Systems Biology: Design Principles of Biological Circuits.* Chapman & Hall, 2006.

[2] M. Amos. *Theoretical and Experimental DNA Computation.* Natural computing series. Springer-Verlag, Berlin, Heidelberg, New York, 2005.

[3] L. Bernardinello and F. de Cindio. A survey of basic net models and modular net classes. In G. Rozenberg, editor, *Advances in Petri Nets: The DEMON Project*, volume 609 of *Lecture Notes in Computer Science*, pages 304–351. Springer-Verlag, Berlin, Heidelberg, New York, 1992.

[4] F. Boogerd, F. J. Bruggeman, J.-H. S. Hofmeyr, and H.V. Westerhoff, editors. *Systems Biology: Philosophical Foundations.* Elsevier Science, 2007.

[5] S. Donatelli and J. Kleijn, editors. *Application and Theory of Petri Nets*, volume 1639 of *Lecture Notes in Computer Science.* Springer-Verlag, Berlin, Heidelberg, New York, 1999.

[6] P. Frisco. P systems, Petri nets, and Program machines. In R. Freund, G. Lojka, M. Oswald, and G. Păun, editors, *Membrane Computing. 6^{th} International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers*, volume 3850 of *Lecture Notes in Computer Science*, pages 209–223. Springer-Verlag, Berlin, Heidelberg, New York, 2006.

[7] P. Frisco. *Computing with Cells. Advances in Membrane Computing.* Oxford University Press, 2008. to appear.

[8] P. Frisco. A hierarchy of computational processes. Technical report, Heriot-Watt University, 2008. HW-MACS-TR-0059 `http://www.macs.hw.ac.uk:8080/techreps/index.html`.

[9] M. Hack. *Petri Net Language*. MIT-Cambridge, MA, 1976.

[10] J. E. Hopcroft and D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[11] M. Jantzen. Language theory of Petri nets. In *Advances in Petri nets 1986, part I on Petri nets: central models and their properties*, pages 397–412. Springer-Verlag, Berlin, Heidelberg, New York, 1987.

[12] H. Kitano. *Foundations of Systems Biology*. MIT press, 2002.

[13] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

[14] D. Noble. *The Music of Life: Biology Beyond the Genome*. Oxford University Press, 2006.

[15] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Rheinisch-Westfälisches Institut für Instrumentelle Mathematik an der Universität Bonn, 1962. Germany, Schrift Nr. 2 (in German).

[16] G. Păun. Computing with membranes. *Journal of Computer and System Science*, 1(61):108–143, 2000.

[17] G. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing: New Computing Paradigms*. Springer Verlag, Berlin, Heidelberg, New York, September 1998.

[18] W. Reisig. *Petri Nets: An Introduction*, volume 4 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer-Verlag, Berlin, Heidelberg, New York, 1985.

[19] W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*, volume 1491 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, Heidelberg, New York, 1998.