Networks of Cells and Petri Nets

Francesco Bernardini[1], Marian Gheorghe[2],
Maurice Margenstern[3], Sergey Verlan[4]

[1] Leiden Institute of Advanced Computer Science, Universiteit Leiden
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
bernardi@liacs.nl
[2] Department of Computer Science, The University of Sheffield
Regent Court, Portobello Street, Sheffield S1 4DP, UK
M.Gheorghe@dcs.shef.ac.uk
[3] Université Paul Verlaine - Metz, UFR MIM, LITA, EA 3097
Ile du Saulcy, 57045 Metz Cédex, France
margens@univ-metz.fr
[4] LACL, Département Informatique, Université Paris 12
61 av. Général de Gaulle, 94010 Créteil, France
verlan@univ-paris12.fr

Summary. We introduce a new class of P systems, called networks of cells, with rules allowing several cells to simultaneously interact with each other in order to produce some new objects inside some other output cells. We define different types of behavior for networks of cells by considering alternative strategies for the application of the rules: sequential application, free parallelism, maximal parallelism, locally-maximal parallelism and minimal parallelism. We devise a way for translating network of cells into place-transition nets with localities (PTL-nets, for short) - a specific class of Petri nets. Then, for such a construction, we show a behavioral equivalence between network of cells and corresponding PTL-nets only in the case maximal parallelism, sequential execution, and free parallelism, whereas we observe that, in the case of locally-maximal parallelism and minimal parallelism, the corresponding PTL-nets are not always able to mimic the behavior of network of cells. Also, we address the reverse problem of finding a corresponding network of cells for a given PTL-net by obtaining similar results concerning the relationships between their semantics. Finally, we present network-of-cells-based models of two classical synchronization problems: producer/consumer and dining philosophers.

1 Introduction

Membrane computing is an emerging branch of natural computing which deals with distributed and parallel computing devices of a bio-inspired type, which are called membrane systems or P systems (see [17], [18], and also [1] for a comprehensive bibliography). P systems, originally devised by Gh. Păun in [17], are introduced as computing devices which abstract from the structure and functioning

of living cells - they are defined as a hierarchical arrangement of regions delimited by membranes (membrane structure), with each region having associated a multiset of objects and a finite set of rules. Rules typically encode mechanisms for consuming/producing objects (evolution) and mechanisms for moving objects across the membranes (communication). For consuming/producing objects, multiset rewriting (i.e., replacing a multiset with another one) is the most generally used mechanism, whereas, for communication, various mechanisms with different biological inspiration have been proposed such as: targets $here, in, out$ [18], symport/antiport [18], conditional uniport [25], boundary rules [4], and carriers [16]. In particular, symport/antiport, conditional uniport and boundary rules introduce in P systems the concept of coupled transport: communication is achieved through cooperation between two or more objects possibly placed in two distinct regions - the inside and the outside of a membrane. The class of P systems was later extended to tissue P systems [15, 19] – a variant of P systems where the underlying structure is defined as an arbitrary graph. Nodes in the graph represent cells which are able to communicate objects alongside the edges of this graph. Specifically, for this communication, the mechanisms of symport/antiport and conditional uniport are transferred to tissue P systems [18, 25] so to have models where communication is achieved through interactions between two neighboring cells (i.e., two cells which are directly connected by means of an edge in the underlying graph). Moreover, it is possible to have tissue P systems where a cell receives objects from a neighboring cell non-deterministically chosen [3], or where a cell produces signals which are replicated and simultaneously sent to all neighboring cells [15].

The work done in [7, 13, 12] then showed that the aforementioned features of transformation and communication in P systems can be interpreted as transitions in place-transition nets (PT-nets, for short) – a specific class of Petri nets (e.g., see [21, 22, 8, 11]). This is done by mapping each rule into a transition with places corresponding to the left-hand side of the rule as input, and with places corresponding to the right-hand side of the rule as output; each place in fact represent the occurrence of a certain object inside a certain membrane. Thus, production/consumption of objects and movement of objects across the membranes are both reflected into modifications on the distributions of tokens inside the places of a PT-net. Specifically, this construction was initially applied in [7] to P systems with boundary rules - which encompass symport/antiport and conditional uniport too - and then re-used in [13, 12] for the basic model of P systems where communication is controlled by targets $here, in, out$. Moreover, contributions [13, 12] devise a formal framework for describing the behavior of P systems in terms of causality/concurrency and for reasoning about reachability, conflicts and soundness of these systems by starting from their translation into PT-nets; this is done by using the PT-net representation of a membrane system is therefore to define the semantics of these systems in terms of sequences of events which consume some resources in order to produce some new ones (process semantics). This construction, which is a standard asynchronous PT-net, is extended in [12, 13] to PT-nets operating in a maximally parallel way and to PT-nets with localities operating in a locally-

maximal parallel way. PT-nets with localities are a class of PT-nets introduced in [13] where each transition belongs to certain location, in a way that resembles the distribution of the rules over the various regions of a membrane system. This makes possible to distinguish between the globally and locally synchronous behavior (maximal parallelism), globally asynchronous but locally synchronous behavior (locally-maximal parallelism), and asynchronous behavior.

In this paper, we introduce a new class of P systems which we call networks of cells. Network of cells are characterized by rules which allow several cells to simultaneously interact with each other in order to produce some new objects inside some other output cells. They are motivated by the observation made in [24] that basic forms of coupled transport like symport/antiport can be expressed in terms of two cells synchronizing on certain inputs in order to produce some outputs. In this respect, networks of cells are not limited to have only two synchronizing cells on their left-hand side and two output cells on their right-hand side. Then, similarly to what was done [13, 12], we define different types of behavior for networks of cells by considering different strategies for the application of the rules: sequential application, free parallelism, maximal parallelism and locally-maximal parallelism plus minimal parallelism [6]. Next, we extend to networks of cells the construction devised in [7, 13, 12] for translating membrane systems into PT-nets with localities (PTL-net, for short) by showing that interaction rules of networks of cells can still be represented as transitions of PTL-nets. However, we are able to establish a behavioral equivalence between network of cells and corresponding PTL-nets only in the case maximal parallelism, sequential execution, and free parallelism (i.e., only for globally and locally synchronous behavior, and asynchronous behavior) as we observe that, in the case of locally-maximal parallelism and minimal parallelism, the corresponding PTL-net is not always able to mimic the behavior of the original network of cells. This allows us to point out differences between the concept of cells used in our model and that of locality introduced in [13] for PT-nets. Also, we address the reverse problem of finding a corresponding network of cells for a given PTL-net by obtaining similar results concerning the relationships between their semantics. Finally we present network-of-cells-based models of two classical synchronization problems: producer/consumer and dining philosophers. These are devised by starting from existing PT-net solutions with the aim of illustrating differences in the two modeling approaches.

2 Preliminaries

We recall some basic notions concerning strings and multisets (e.g., see [23, 18] for further details).

An *alphabet* is any finite and non-empty set. The elements of an alphabet are called *symbols*. Let V be an alphabet. A *string* over V is any finite sequence consisting of zero or more symbols from V; the same symbol may occur repeated several times inside the same string. The sequence containing no symbols is called

empty string and it is denoted by λ. The set of all strings (respectively of all non-empty strings) over V is denoted by V^* (respectively V^+). If $x, y \in V^*$, then their *catenation* is $xy \in V^*$ (the catenation of two strings is the string obtained by the juxtaposition of the two strings, that is, by writing one string after the other one). Catenation is an associative operation and the empty string λ acts as an identity: $x\lambda = \lambda x = x$, for all $x \in V^*$. Moreover, for any $i \geq 1$, we denote by x^i the catenation of i copies of the string x; we set $x^0 = \lambda$ by definition. The *length of a string* $x \in V^*$, denoted by $|x|$, is the number of all occurrences in x of symbols from V; the number of occurrences in x of a symbol $a \in V$ is denoted by $|x|_a$. Yet again, by definition, we set $|\lambda| = 0$ and $|\lambda|_a = 0$, for all $a \in V$. The set of symbols from V occurring in a string x is denoted by $\aleph(x)$. We also use V to denote the set of strings from V^* of length equal to 1.

Let V be an alphabet. A *(finite) multiset (over V)* is a mapping $M : V \longrightarrow \mathbb{N}$, where $\mathbb{N}$ denotes the set of natural numbers (0 included); for every $a \in V$, $M(a)$ is called the *multiplicity of a (in M)*. A multiset over V is usually given in the form of a string over V; each string $x \in V^*$ in fact identifies a multiset M such that, for every $a \in V$, $M(a) = |x|_a$. On the other hand, every multiset M over V is representable by means of any string $x \in V^*$ such that, for every $a \in V$, $|x|_a = a$ – the order of the symbols in such a string is not important. Therefore, from this moment on, we will use strings to represent multisets and, given $x \in V^*$, we will use the expression "multiset x" to refer to a multiset representable by means of string x. Thus, for $x \in V^*$, for $a \in V$, the multiplicity of a in x is $|x|_a$, and the *size of multiset* x (i.e., the sum of all multiplicities) is the value $|x|$. Moreover, for $x, y \in V^*$, we say that multiset x and multiset y are equal, and we write $x = y$, if, for all $a \in V$, $|x|_a = |y|_a$. We say that multiset x *includes* multiset y, and we write $x \sqsupseteq y$, if, for all $a \in V$, we have $|x|_a \geq |y|_a$. If that is the case, we also say that y *is included* in x, and we write $y \sqsubseteq x$. The union of multiset x and multiset y, denoted by $x \sqcup y$, is a multiset w such that, for all $a \in V$, $|w|_a = |x|_a + |y|_a$.

The notion of a rewriting rule between strings can be naturally transferred to multisets. A *multiset rewriting rule* is a pair (u, v), with u, v two multisets, which is written in the form $u \rightarrow v$. Given a multiset w and a rewriting rule $u \rightarrow v$, if $u \sqsubseteq w$, then the rule $u \rightarrow v$ is *applicable* to the multiset w; if that is the case, the result of the application of the rule $u \rightarrow v$ to the multiset w is the multiset w' such that, for all $a \in V$, $|w'|_a = |w|_a - |u|_a + |v|_a$. If that is case, then we also say that the multiset w *can evolve* by means of the multiset rewriting rule $u \rightarrow v$.

3 Networks of Cells

Here we introduce a general model of membrane systems which allows us to capture the essential features of most variants of cell-like P systems and tissue P systems.

Definition 1 (network of cells). *A* network of cells of degree $n \geq 1$ *(an* NC of degree $n \geq 1$*, for short) is a construct:*

$$\Pi = (V, w_1, w_2, \ldots, w_n, R),$$

where:

1. *V is an* alphabet*;*
2. *$w_i \in O^*$, for all $1 \leq i \leq n$, is the multiset* initially associated to cell *i;*
3. *R is a finite set of* interaction rules *of the form*

$$(u_1, i_1) \dots (u_k, i_k) \rightarrow (v_1, j_1) \dots (v_h, j_h)$$

where (a) $1 \leq k, h \leq n$, (b) for all $1 \leq l, l' \leq k$, $u_l \in V^+$, (c) $1 \leq i_l \leq n$ and $l \neq l'$ implies $i_l \neq i_{l'}$, (d) for all $1 \leq r, r' \leq h$, $v_r \in V^*$, $1 \leq j_r \leq n$, and $r \neq r'$ implies $j_r \neq j_{r'}$.

A network of cells consists of n cells numbered from 1 to n with each one of them containing a multiset of objects over V (initially cell i contains multiset w_i). Cells can interact with each other by means of the rules in R. An interaction rule of the form $(u_1, i_1) \dots (u_k, i_k) \rightarrow (v_1, j_1) \dots (v_h, j_h)$ specifies that, whenever, at the same time, for all $1 \leq l \leq k$, cell i_l contains at least one occurrence of multiset u_l, an occurrence of the multiset u_l is consumed inside cell i_l, for all $1 \leq l \leq k$, and a multiset v_r is produced inside cell j_r, for all $1 \leq r \leq h$. In other words, an interaction rule simultaneously rewrites some multisets inside cells $i_1, \dots, i_k$ in order to produce some new multisets inside cells $j_1, \dots, j_h$. Notice also that, for an interaction rule $(u_1, i_1) \dots (u_k, i_k) \rightarrow (v_1, j_1) \dots (v_h, j_h)$, for all $1 \leq l, l' \leq k$, we have $u_l \in V^+$ (i.e., a multiset on the left-hand side of the rule cannot be empty) and $l \neq l'$ implies $i_l \neq i_{l'}$ (i.e., the left-hand side of the rule must involve a set of distinct cells), and, for all $1 \leq r, r' \leq h$, we have $v_r \in V^*$ (i.e., a multiset on the right-hand side of the rule can be empty) and $r \neq r'$ implies $j_r \neq j_{r'}$ (i.e., the right-hand side of the rule must involve a set of distinct cells).

For an interaction rule ρ of the form $(u_1, i_1) \dots (u_k, i_k) \rightarrow (v_1, j_1) \dots (v_h, j_h)$, cells $i_1, \dots, i_k$ are called *input cells*, the set $\{i_1, \dots, i_k\}$ is denoted by $Input(\rho)$ and k is called the *input radius of* ρ; cells $j_1, \dots, j_h$ are called *output cells*, the set $\{j_1, \dots, j_h\}$ is denoted by $Output(\rho)$ and h is called the *output radius of* ρ; the *cooperation degree of* ρ is the value $\max\{|u_i| \mid 1 \leq i \leq k\}$; the left-hand side $(u_1, i_1) \dots (u_k, i_k)$ is denoted by $lhs(\rho)$ whereas the right-hand side $(v_1, j_1) \dots (v_h, j_h)$ is denoted by $rhs(\rho)$. Also, for such a rule ρ, for all $1 \leq i \leq n$, we use $lhs_i(\rho)$ to denote the multiset u if $(u, i) \in lhs(\rho)$, or λ if $i \notin Input(\rho)$; we use $rhs_i(\rho)$ to denote the multiset v if $(v, i) \in rhs(\rho)$, or λ if $i \notin Output(\rho)$.

Notice that the structure of an NC corresponds neither to a tree as in cell-like P systems nor to a graph as in tissue P systems (e.g., see [18] for definitions of cell-like P systems and tissue P systems), though some models of cell-like P systems and tissue P systems can be seen as special variants of NC's. The possibility of representing existing variants of P systems as NC's is illustrated through the following examples.

Example 1. A basic P system [18] is defined as a hierarchical arrangement of membranes; each membrane delimits a region which contains a multiset of objects and finite set of evolution rules. If V is the alphabet of the system, then an evolution

rule has the form $u \to (u_1, t_1)(u_2, t_2) \dots (u_q, t_q)$ with $u \in V^*$, for all $1 \leq r \leq q$, $u_r \in V^*$ and $t_r \in \{here, out, in_j\}$. If such a rule is associated to a region i, then multiset u can be replaced by multisets $u_1, u_2, \dots, u_q$ and each multiset u_r is moved across the membranes depending on the target t_r: u_r remains inside membrane i when $t_r = here$, u_r is moved outside membrane i when $t_r = out$, u_r is moved into region j when $t_r = in_j$ with j a membrane directly contained into region i.

A basic P system can be represented as an NC which has as many cells as the membranes in the P system and which contains, for every region i of the P systems, for every evolution rule $u \to (u_1, t_1)(u_2, t_2) \dots (u_q, t_q)$ associated to region i, an interaction rule $(u, i) \to (u_1, j_1)(u_2, j_2) \dots (u_q, j_q)$, with distinct $j_1, j_2, \dots, j_q$, such that, for all $1 \leq r \leq q$, if $t_r = here$, then $j_r = i$; if $t_r = in_j$, then $j_r = j$; if $t_r = out$, then j_r is equal to the index of the directly upper region.

Example 2. P systems with boundary rules [4] extend basic P systems with rules which allow direct interactions between the inside and the outside of a region. In their most general form (e.g., see [5]), boundary rules are of the form $u \lfloor_i v \to u' \lfloor_i v'$ with i the index of a membrane in the system and $u, v, u', v' \in V^*$, for V the alphabet of the system.

These rules can be represented in NC's as interactions of the form $(u, j)(v, i) \to (u', j)(v', i)$ with j the membrane which directly contains membrane i. In this way, we can for instance capture the features of symport/antiport rules [18]:

- an antiport rule $(x, in; y, out)$ associated to membrane i is no more than an interaction rule $(x, j)(y, i) \to (y, j)(x, i)$ with j the membrane which directly contains membrane i;
- a symport rule (x, out) associated to membrane i is no more than an interaction rule $(x, i) \to (x, j)$ with j the membrane which directly contains membrane i;
- a symport rule (x, in) associated to membrane i is no more than an interaction rule $(x, j) \to (x, i)$ with j the membrane which directly contains membrane i.

Example 3. An evolution-communication model of tissue P systems is proposed in [3] that is based on graphs of cells where each cell contains a multiset of objects and a finite set of rules of the forms $x \to y$ (transformation rules) and $(x; y, in)$ (communication rules) with x, y two multisets over a given alphabet. A transformation rule $x \to y$ associated to a cell i specifies that a multiset x placed inside cell i can be replaced by a multiset y which remains inside cell i. A communication rule $(x; y, in)$ associated to a cell i instead specifies that, in presence of a multiset x, a multiset y can be moved from a neighboring cell j non-deterministically chosen into cell i.

Such a tissue P system can be represented as an NC with the same number of cells which contains an interaction rule $(x, i) \to (y, i)$, for every transformation rule $x \to y$ associated to cell i of the tissue P system, and a set of interaction rules $\{(x, i)(y, j) \to (xy, j) \mid \{i, j\}$ is an edge of the graph underlying the tissue P system$\}$, for every communication rule $(x; y, in)$ in the tissue P system associated to cell i.

We now pass to precisely define the execution semantics of networks of cells by identifying different strategies for the application of the rules. To this aim, we first give the following definitions.

Definition 2 (configuration). *Let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells. A* configuration *of Π is any tuple $(w'_1, \ldots, w'_n)$ with $w_i \in O^*$, for all $1 \leq i \leq m$. The* initial configuration of Π *is the tuple $(w_1, \ldots, w_n)$.*

Definition 3 (multiset of applicable rules). *Let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$ be a configuration of Π. A* multiset of applicable rules (w.r.t. Π and C) *is any function $\Gamma_C : R \to \mathbb{N}$ such that, for all $1 \leq i \leq n$, $(\bigsqcup_{r \in R}(lhs_i(r))^{\Gamma_C(r)}) \sqsubseteq w'_i$.*

Free parallelism

Free parallelism means that, in each step, any multiset of applicable rules can be used to make an NC transit from a configuration to another one by applying all the selected rules in parallel at the same time. In membrane computing literature, this semantics is also called asynchronous behavior (e.g., see [9]).

Specifically, let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$, $C' = (w''_1, \ldots, w''_n)$ be two configurations of Π. We say that Π transits in one step from configuration C to configuration C' in a freely-parallel way, and we write $C \Rightarrow_{free} C'$, if there is a multiset of applicable rules Γ_C such that, for all $1 \leq i \leq n$, $w''_i = (w'_i \setminus (\bigsqcup_{r \in R}(lhs_i(r))^{\Gamma_C(r)})) \cup (\bigsqcup_{r \in R}(rhs_i(r))^{\Gamma_C(r)})$.

Thus, in a freely-parallel step, an arbitrary multiset of applicable rules is selected and these rules are applied in parallel by consuming all the multisets on their left-hand sides and producing all the multisets on their right-hand sides in the respective places.

Sequential execution

Sequential execution means that, in each step, only one rule is applied to make an NC transit from a configuration to another one.

Specifically, let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$, $C' = (w''_1, \ldots, w''_n)$ be two configurations of Π. We say that Π transits in one step from configuration C to configuration C' in a sequential way, and we write $C \Rightarrow_{seq} C'$, if $C \Rightarrow_{free} C'$ for some multiset of applicable rules Γ_C with $|\Gamma_C| = 1$.

Thus, a sequential step is a freely-parallel step where the number of rules used is equal to 1.

Maximal parallelism

Maximal parallelism means that, in each step, any maximal multiset of applicable rules can be used to make an NC transit from a configuration to another one by

applying all the selected rules in parallel at the same time; a maximal multiset of applicable rules is any multiset of applicable rules to which no other rules can be added so to obtain another multiset of applicable rules. Maximal parallelism is the type of behavior which was associated to membrane systems in their original definition [17], and it is the semantics most commonly adopted in the area of membrane computing (e.g., see [1], [18]).

Specifically, let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$, $C' = (w''_1, \ldots, w''_n)$ be two configurations of Π. We say that Π transits in one step from configuration C to configuration C' in a maximally-parallel way, and we write $C \Rightarrow_{max} C'$, if $C \Rightarrow_{free} C'$ for some multiset of applicable rules Γ_C such that, for all $r \in R$, there is $1 \leq i \leq n$ with $lhs(r)_i \neq \lambda$ and $lhs_i(r) \not\sqsubseteq (w'_i \setminus (\bigsqcup_{r \in R}(lhs_i(r))^{\Gamma_C(r)}))$.

Thus, a maximally-parallel step is a freely-parallel step where rules are applied in parallel in an exhaustive way: once the multisets on the left-hand side of these rules are consumed, no other rule has to be applicable to the objects left inside the cells.

Locally-maximal parallelism

Locally-maximal parallelism, which was introduced in [13], specifies that, in each step, if a cell is involved in the application of (at least) one rule, then a maximal number of objects are consumed in this cell by applying in parallel as many rules that involve this cell as possible.

Specifically, let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$, $C' = (w''_1, \ldots, w''_n)$ be two configurations of Π. We say that Π transits in one step from configuration C to configuration C' in a locally-maximally-parallel way, and we write $C \Rightarrow_{lmax} C'$, if $C \Rightarrow_{free} C'$ for some multiset of applicable rules Γ_C such that, for all $1 \leq i \leq n$, if there is $r' \in R$ with $\Sigma_C(r') > 0$ and $lhs_i(r') \neq \lambda$, then, for all $r \in R$ with $lhs(r)_i \neq \lambda$, there is $1 \leq j \leq n$ with $lhs(r)_j \neq \lambda$ and $lhs_j(r) \not\sqsubseteq (w'_j \setminus (\bigsqcup_{r \in R}(lhs_j(r))^{\Gamma_C(r)}))$.

Thus, a locally-maximally parallel step is a freely-parallel step where if a cell is involved in the application of one rule, then a maximal number of rules involving this cell is applied in parallel at the same time. In other words, in a locally-maximally parallel step, every cell that gets involved tries to participate in as many interactions as possible depending on the objects currently available inside the cell and on the presence of other cells competing for the same objects.

Minimal parallelism

Minimal parallelism, which was introduced in [6], means that, in each step, every cell that can participate in at least one interaction must get involved in the application of at least one rule.

Specifically, let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells and let $C = (w'_1, \ldots, w'_n)$, $C' = (w''_1, \ldots, w''_n)$ be two configurations of Π. We say that

Π transits in one step from configuration C to configuration C' in a minimally-parallel way, and we write $C \Rightarrow_{min} C'$, if $C \Rightarrow_{free} C'$ for some multiset of applicable rules Γ_C such that, for all $1 \leq i \leq n$, if there is no $r' \in R$ with $\Gamma_C(r') > 0$ and $lhs_i(r') \neq \lambda$, then, for all $r \in R$ with $lhs(r)_i \neq \lambda$, there is $1 \leq j \leq n$ with $lhs(r)_j \neq \lambda$ and $lhs_j(r) \not\sqsubseteq (w'_j \setminus (\bigsqcup_{r \in R}(lhs_j(r))^{\Gamma_C(r)}))$.

Thus, a minimally-parallel step is a freely-parallel step where a maximal number of cells, which are selected depending on the current distribution of objects inside the cells, evolve in parallel at the same time; each one of the selected cells has to participate in at least one interaction, and no other rule has to be applicable in parallel at the same time that involve any cell different from the selected ones.

Example 4. Let $UNO = (V, aa, aa, bb, c, R)$ be an NC where R contains the following interaction rules:

1. $(a,1) \rightarrow (a,1)(a,2)$,
2. $(a,1) \rightarrow (a,1)(b,3)$,
3. $(c,4)(a,2)(b,3) \rightarrow (b,2)(a,3)$,
4. $(b,2) \rightarrow (b,2)(c,4)$,
5. $(c,4)(b,3) \rightarrow (cc,4)(b,3)$,
6. $(c,4)(b,2) \rightarrow (bb,3)$,
7. $(c,4)(a,1) \rightarrow (a,4)(a,2)(a,3)$.

This NC has rules with input radius at most 2 and cooperation degree 1. The initial configuration is given by the tuple $C_0 = (aa, aa, bb, c)$. To the initial configuration, we can apply rule 1 with multiplicity at most 2, rule 2 with multiplicity at most 2, rule 3 with multiplicity at most 1, rule 5 with multiplicity at most 1, and rule 7 with multiplicity at most 1. However, with respect to C_0, it is not possible to apply all these rules in parallel (e.g., only one rule between rules 3, 5 and 7 can be applied because cell 4 contains only one object c).

Thus, in the case of free-parallelism, for any configuration C' obtained by applying any combination of the aforementioned rules that is a multiset of applicable rules, we have $C_0 \Rightarrow_{free} C'$. For instance, if rule 1 is applied with multiplicity 2 and rule 3 is applied with multiplicity 1, we have $C_0 \Rightarrow (aa, aaab, ab, \lambda)$.

In the case of sequential execution, only one of the aforementioned rules is going to be applied with multiplicity 1 to the initial configuration. This gives us five possible transitions: $C_0 \Rightarrow_{seq} (aa, aaa, bb, c)$ when rule 1 is applied, $C_0 \Rightarrow_{seq} (aa, aa, bbb, c)$ when rule 2 is applied, $C_0 \Rightarrow_{seq} (aa, ab, ab, \lambda)$ when rule 3 is applied, $C_0 \Rightarrow_{seq} (aa, aa, bb, cc)$ when rule 5 is applied and $C_0 \Rightarrow_{seq} (a, aaa, abb, a)$ when rule 7 is applied.

If maximal parallelism is adopted, then a maximal number of the aforementioned rules is applied to the initial configuration C_0. Specifically, we have the following possibilities: rule 1 (rule 2) applied with multiplicity 2 in parallel with another rule chosen between rules 3 and 5; rule 1 applied in parallel with rule 2 (both with multiplicity 1) together with another rule chosen between rules 3 and

5; rule 1 (rule 2) applied with multiplicity 1 in parallel with rule 7. For instance, if we choose this latter combination, we have $C_0 \Rightarrow_{max} (a, aaaa, abb, a)$.

In the case of locally-maximal parallelism, we have to make sure that if a cell evolves by means of at least one rule, then all other rules affecting that same cell that can be applied in parallel are effectively applied within the same step of execution. Specifically, for the initial configuration C_0, this means that whenever rule 1, or 2 (rule 7) is used, then another rule chosen between rules 1, 2 and 7 (rules 1 and 2) is always applied in parallel in the same step. On the other hand, we have $C_0 \Rightarrow_{lmax} (aa, ab, ab, \lambda)$ by just applying rule 3 because there are no other rules involving cell 2, or 3 or 4 that can be applied in parallel at the same time. Also, we have $C_0 \Rightarrow_{seq} (aa, aa, bb, cc)$ by just applying rule 5 because there are no other rules involving cell 3 or 4 that can be applied in parallel at the same time.

In the case of minimal parallelism, a maximal number of cells evolve in parallel at the same time with each one of the selected cells participating in at least one interaction. Specifically, for the initial configuration C_0, this means that rule 3 (rule 5) is always used in parallel with at least an application of rule 1 or 2. However, with respect to C_0, we have $C_0 \Rightarrow_{min} (a, aaa, abb, aa)$ by just applying rule 7 because there are no rules involving cells 2 or 3 which can be applied in parallel at the same time.

Remark 1. Locally-maximal parallelism was introduced in [13, 12] only for the basic model of P systems where rules are precisely assigned to regions delimited by membranes and the left-hand side of every rule involves only objects inside the region which the rule is assigned to. Therefore, locally-maximal parallelism is defined in [13, 12] by simply stating that, in each step, a certain number of membranes is selected and a maximal number of rules is applied inside each one of these membranes. In the case of NC's, interaction rules may involve objects placed inside different cells, hence locally-maximal parallelism is defined with respect to a certain group of cells: in each step, a multiset of applicable rules can be used only if it is maximal with respect to the cells which appear on the left-hand side of the rules in it. However, if the rules of NC's are restricted to have input radius equal to 1, then the present notion of locally-maximal parallelism is consistent with the semantics given in [13, 12] for the basic model of P systems.

Remark 2. The minimally-parallel semantics for NC's is not defined in terms of number of rules which are applied inside the cells, as in [6], but it is defined with respect to the number of cells which can evolve in parallel at the same time. Specifically, in each minimally-parallel step, a multiset of applicable rules can be used only if there are no cells which do not appear on the left-hand side of any rule in it and which some rules can be applied to; the multiset of applicable rules has not to be maximal neither locally nor globally though. Minimal parallelism was instead defined in [6] for P systems with symport/antiport where every membrane has its own set of rules, hence, in each step, for each one of these sets of rules, if there is a rule which is applicable, then at least one rule from that set is going to be applied irrespective of the fact that an antiport rule involves two distinct

regions at the same time. Therefore, although symport/antiport can be expressed as interaction rules of NC's, the notion of minimal parallelism proposed in [6] for symport/antiport differs from the one considered here because interaction rules of NC's are not assigned a priori to any cell. However, if rules of NC's are restricted to have input radius equal to 1, then our minimally-parallel semantics for NC's is consistent with the idea from [6], that is, in each step, if at least one rule may be used inside a region, then at least one rule is applied inside that region.

Remark 3. From a computational point of view (in the usual sense of membrane computing), if we consider NC's operating according to maximal parallelism, then it is obvious that they are computationally complete and that the hierarchy on the number of cells collapses at level 1. Moreover, the universality results obtained for catalytic P systems and evolution-communication P systems (e.g., see [14], [3], [10]) can be directly transferred to NC's: NC's with rules of input radius at most 2 are computationally complete and, for such systems, the hierarchy on the number of cells collapses at level 2. More precisely, for NC's corresponding to catalytic P systems, we have universality for input radius at most 1 and cooperation degree at most 2 [10], whereas, for evolution-communication P systems, we have universality for input radius at most 2 and cooperation degree at most 1 when antiport rules are used, or for input radius at most 1 and cooperation degree at most 2 when symport rules are used [14]. On the other hand, the computational power of NC's operating in a sequential manner, in a freely-parallel manner, in a locally-maximal parallel manner, or in a minimally parallel manner requires further investigations.

4 PT Nets with Localities

We introduce the class of Petri nets called place-transition nets with localities in the form reported in [12].

Definition 4 (PTL-net). *A* PT-net with localities *(a* PTL-net*, for short) is a construct:*

$$N = (P, T, W, M_0, L),$$

where

1. *P is a finite set of symbols whose elements are called* places*,*
2. *T is a finite set of symbols whose elements are called* transitions*,*
3. *$W : (P \times T) \cup (T \times P) \to \mathbb{N}$ is the* weight function*,*
4. *$M_0 \in P^*$ is a multiset over P called the* initial marking*,*
5. *$L : T \to \mathbb{N}$ is a* locality mapping*;*

and such that $P \cap T = \emptyset$.

PTL-nets are usually represented by diagrams where places are drawn as circles, transitions are drawn as squares, and a directed arc (x, y) is added between x and y if $W(x, y) \geq 1$. Moreover, transitions are annotated with their localities and the

arcs are annotated with their weights if these are 2 or more. Localities are used to partition the set of transitions into subsets of transitions which logically belongs to distinct locations.

Given a PTL-net N, the *pre-* and *post-multiset* of a transition t are respectively the multiset $pre_N(t)$ and the multiset $post_N(t)$ such that, for all $p \in P$, $|p|_{pre_N(t)} = W(p,t)$ and $|p|_{post_N(t)} = W(t,p)$. A configuration of N, which is called a *marking*, is any multiset over P; in particular, for every $p \in P$, $|p|_M$ represents the number of *tokens* present inside place p. Then, we recall from [12] the notion of an execution mode for a PTL-net by giving the following definitions.

Definition 5 (free-enabled). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let M be a marking of N. A multiset of transitions $U \in T^*$ is* free-enabled *at marking M if $(\bigsqcup_{t \in T}(pre_N(t))^{|t|_U}) \sqsubseteq M$.*

Definition 6 (seq-enabled). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let M be a marking of N. A multiset of transitions $U \in T^*$ is* seq-enabled *at marking M if U is free-enabled at marking M and $|U| = 1$.*

Definition 7 (max-enabled). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let M be a marking of N. A multiset of transitions $U \in T^*$ is* max-enabled *at marking M if U is free-enabled at marking M and, for all $t \in T$, $pre_N(t) \not\sqsubseteq (M \setminus (\bigsqcup_{t \in T}(pre_N(t))^{|t|_U}))$.*

Definition 8 (lmax-enabled). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let M be a marking of N. A multiset of transitions $U \in T^*$ is* lmax-enabled *at marking M if U is free-enabled at marking M and, for all $l \in L(T)$, if there is $t \in T$ with $L(t) = l$ and $|U|_t > 0$, then, for all $t' \in T$ with $L(t') = l$, $pre_N(t') \not\sqsubseteq (M \setminus (\bigsqcup_{t \in T}(pre_N(t))^{|t|_U}))$.*

Definition 9 (min-enabled). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let M be a marking of N. A multiset of transitions $U \in T^*$ is* min-enabled *at marking M if U is free-enabled at marking M and, for all $l \in L(T)$, if there is no $t \in T$ with $L(t) = l$ and $|U|_t > 0$, then, for all $t' \in T$ with $L(t') = l$, $pre_N(t') \not\sqsubseteq (M \setminus (\bigsqcup_{t \in T}(pre_N(t))^{|t|_U}))$.*

Definition 10 (m-execution). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let* m $\in \{free, seq, max, lmax, min\}$. *The* m-execution *of N is the relationship $\rightsquigarrow_{\mathrm{m}} \subseteq P^* \times P^*$ such that, for all $M, M' \in P^*$, $M \rightsquigarrow_{\mathrm{m}} M'$ iff, $M' = (M \setminus (\bigsqcup_{t \in T}(pre_N(t))^{|t|_U}))$ for some $U \in T^*$ which is* m*-enabled at marking M.*

Thus, an m-execution of a PTL-net N represents a transition step which makes possible to derive a new marking from a given one by firing a certain number of transitions in parallel at the same time. The firing of each transition results in the consumption of its pre-multiset of places from the given marking and in the production of its post-multiset of places in the new marking; the number of transitions that can fire in parallel within a transition step depends on the execution

mode m $\in \{free, seq, max, lmax, min\}$ and it has to be consistent with the current availability of tokens inside each place (i.e., for each place, the number of its tokens consumed cannot be greater than its multiplicity in the current marking). Specifically, the aforementioned execution modes identify the following behaviors for a PTL-net:

- *free-execution*: in each transition step, an arbitrary number of transitions fire by providing that these constitute a free-enabled multiset of transitions (i.e., the union of their pre-multisets has to be contained in the current marking).
- *seq-execution*: in each transition step, only one transition fires that is chosen amongst those whose pre-multiset of place is contained in the current marking (i.e., in order to fire, a transition has to be enabled at the current marking);
- *max-execution*: in each transition step, a maximal number of transitions fire by providing that these constitute a free-enabled multiset of transitions which no other transition can be added to in order to obtain another free-enabled multiset of transitions (i.e., the selected transitions has to consume a maximal number of places so that no other transition can fire in parallel at the same time);
- *lmax-execution*: in each transition step, an arbitrary set of localities is selected and, for each one of these localities, a maximal number of transitions fire (i.e., for each selected locality, a maximal number of places is consumed so that no other transition belonging to the same locality can fire in parallel at the same time);
- *min-execution*: in each transition step, for each locality such that there is at least one enabled transition associated to that locality, at least one transition fire (i.e., the selected multiset of transitions has to involve a maximal set of localities, although the number of firing transition belonging to the same locality has not necessarily to be maximal with respect to the current marking).

Notice that seq-execution is called min-execution in [13, 12], whereas the present notion of min-execution is introduced as a counterpart of the minimally-parallel semantics previously used in the area of membrane computing [6]; this latter notion of minimal parallelism is in fact not considered in [13, 12].

5 Network of Cells versus PTL-nets

We start by extending to the class of network of cells the basic construction devised in [7, 12, 13] to transform membrane systems into "equivalent" PTL-nets.

Definition 11. *Let $\Pi = (V, w_1, w_2, \ldots, w_n, R)$ be a network of cells.*

1. *The* extended alphabet of Π*, denoted by E_Π, is the set $\{(a, i) \mid a \in V, 1 \leq i \leq n\}$.*
2. *For all $1 \leq i \leq n$, the i* labeling of Π *is the mapping $h_{i,\Pi} : V^* \to E_\Pi^*$ such that, for all $a \in V$, $h_{i,\Pi}(a) = (a, i)$ and, for all $u, v \in V^*$, $h_{i,\Pi}(uv) = h_{i,\Pi}(u)h(v)$.*

3. *For all* $r \in R$ *of the form* $(u_1, i_1) \dots (u_k, i_k) \rightarrow (v_1, j_1) \dots (v_h, j_h)$, *the* cell-labeled version of r, *denoted by* $CL(r)$, *is the (multiset rewriting) rule* $h_{i_1,\Pi}(u_1) \dots h_{i_k,\Pi}(u_k) \rightarrow h_{j_1,\Pi}(v_1) \dots h_{j_h,\Pi}(v_h)$.

Thus, for all $1 \leq i \leq n$, the i labeling assigns to every object of a given multiset the label i; the cell-labeled version of an interaction rule is a multiset rewriting rules where the localization of the multisets inside the cells is given by the labels assigned to the objects. This idea of assigning a label to the objects in order to represent their localization inside the cells is central to the following construction which shows how to define a corresponding PTL-net for every network of cells.

Definition 12 (corresponding PTL-net). *Let* $\Pi = (V, w_1, w_2, \dots, w_n, R)$ *be a network of cells where rules in* R *are labeled in a one-to-one manner with values in* $\{1, 2, \dots, |R|\}$. *The PTL-net corresponding to* Π *is* $\mathcal{N}(\Pi) = (E_\Pi, \{1, 2, \dots, |R|\}, W, M_0, L)$, *where: for all* $p \in E_\Pi$, $t \in \{1, 2, \dots, |R|\}$, $W(p,t) = m$ *iff* t *is the label of a rule* $r \in R$ *with* $m = |lhs(CL(r))|_p$, *and* $W(t,p) = m$ *iff* t *is the label of a rule* $r \in R$ *with* $m = |rhs(CL(r))|_p$; $M_0 = h_{1,\Pi}(w_1)h_{2,\Pi}(w_2) \dots h_{n,\Pi}(w_n)$; *for all* $t \in \{1, 2, \dots, |R|\}$, $L(t) = 0$.

Thus, an NC Π is transformed into a PTL-net which contains a place for each cell in Π and for each object possibly present inside this cell, and a transition for each rule in Π. More precisely, in the corresponding PTL-net, occurrences of the same symbol inside different cells are represented as occurrences of tokens inside different places, each one of them identifying the presence of that symbol inside a certain cell. The consumption of objects from certain places and the production of new objects in other cells are then reflected in the movement of tokens between the respective places; pre- and post-multisets of the transitions in fact correspond to left-hand sides and right-hand sides of the rules in Π respectively.

As an example, we show in Figure 1 the PTL-net corresponding to the NC UNO of Example 4.

Next, similar to what was done in [13, 12], we introduce the notion of equivalence between the behavior of an NC and that of its corresponding PTL-net.

Definition 13 (m-equivalence). *Let* $\Pi = (V, w_1, w_2, \dots, w_n, R)$ *be a network of cells and let* $\mathcal{N}(\Pi)$ *be its corresponding PTL-net. For* $\mathrm{m} \in \{free, seq, max, lmax, min\}$, *we say that* Π *is* m-equivalent to $\mathcal{N}(\Pi)$, *and we write* $\Pi \equiv_\mathrm{m} \mathcal{N}(\Pi)$, *if, for every two configurations* $C = (w'_1, \dots, w'_n)$, $C' = (w''_1, \dots, w''_n)$ *of* Π, $C \Rightarrow_\mathrm{m} C'$ *iff* $h_{1,\Pi}(w'_1) \dots h_{i,n}(w'_n) \rightsquigarrow_\mathrm{m} h_{1,\Pi}(w''_1) \dots h_{i,n}(w''_n)$.

Thus, it is easy to see that the following proposition holds.

Proposition 5.1 *For any network of cells* Π, *for* $\mathrm{m} \in \{free, seq, max\}$, *we have that* $\Pi \equiv_\mathrm{m} \mathcal{N}(\Pi)$.

On the other hand, since all the transitions of the corresponding PTL-net are assigned to the same locality, we have that, for some network of cells Π, $\Pi \not\equiv_{lmax} \mathcal{N}(\Pi)$ and $\Pi \not\equiv_{min} \mathcal{N}(\Pi)$. However, we can think of choosing a different locality

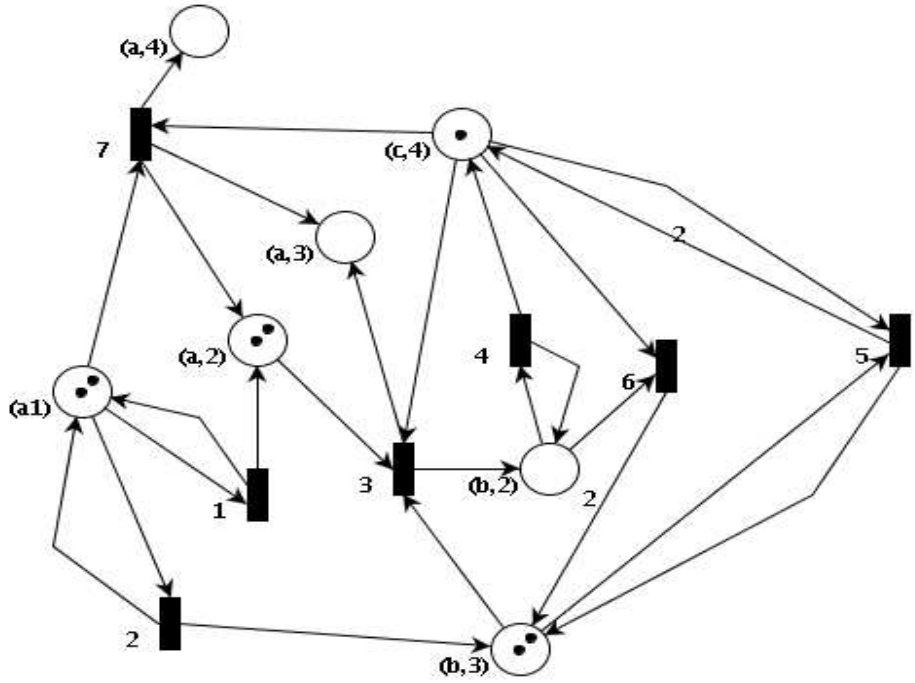

Fig. 1. PTL-net $\mathcal{N}(UNO)$ with its initial marking.

mapping for the construction of Definition 12 in order to establish the equivalence between the locally-maximal (or minimally-parallel) semantics of networks of cells and the locally-maximal (or minimally-parallel) execution of the corresponding PTL-net. To this aim, given an NC $\Pi = (V, w_1, w_2, \ldots, w_n, R)$, and a function $F : \{1, \ldots, |R|\} \to \mathbb{N}$, it is useful to consider the PTL-net $\mathcal{N}(\Pi, F)$ which is constructed as the one of Definition 12 except for the locality mapping which is replaced by F. Thus, we can ask whether, for any NC $\Pi = (V, w_1, w_2, \ldots, w_n, R)$, there exists a $F : \{1, \ldots, |R|\} \to \mathbb{N}$ such that $\Pi \equiv_{lmax} \mathcal{N}(\Pi, F)$ (or $\Pi \equiv_{min} \mathcal{N}(\Pi, F)$), or not.

Proposition 5.2 *There exists a network of cells Π with* 3 *rules such that, for all $F : \{1, 2, 3\} \to \mathbb{N}$, we have that $\Pi \not\equiv_{lmax} \mathcal{N}(\Pi, F)$.*

Proof. Let $DUE = (\{a, b\}, aa, bb, R)$ be an NC where R contains: rule $(a, 1) \to (aa, 1)$ labeled by 1, rule $(a, 1)(b, 2) \to (a, 2)(b, 1)$ labeled by 2, and rule $(b, 2) \to (bb, 2)$ labeled by 3. Then, let us suppose that $DUE \equiv_{lmax} \mathcal{N}(DUE, F)$ for some $F : \{1, 2, 3\} \to \mathbb{N}$.

By Definition 12, PTL-net $\mathcal{N}(DUE, F)$ is the PTL-net (P, T, W, M_0, L) with $P = \{(a, 1), (a, 2), (b, 1), (b, 2)\}$, $T = \{1, 2, 3\}$, $W((a, 1), 1) = 1$, $W(1, (a, 1)) = 2$,

$W((a,1),2) = 1$, $W((b,2),1) = 1$, $W(2,(b,1)) = 1$, $W(2,(a,2)) = 1$, $W((b,2),3) = 1$, $W(3,(b,2)) = 2$, $W(x,y) = 0$ in all other cases, and $M_0 = (a,1)(a,1)(b,2)(b,2)$.

Now, we observe that if $F(1) = F(3)$, then, for the initial configuration C_0 of DUE, $C_0 \Rightarrow_{lmax} (aaaa, bb)$, but $M_0 \not\rightsquigarrow_{lmax} (a,1)(a,1)(a,1)(a,1)(b,2)(b,2)$. This is because if transition 1 and transition 2 are assigned the same locality, then it is not possible to fire transition 1 with multiplicity 2 without firing in parallel transition 3 with multiplicity 2. This contradicts our hypothesis, hence it has to be $F(1) \neq F(3)$ irrespectively of the value of $F(2)$.

Next, if $F(1) \neq F(2)$, then $M_0 \rightsquigarrow_{lmax} (a,1)(b,1)(a,2)(b,2)(b,2)$ by selecting localities $F(2)$ and $F(3)$, but $C_0 \not\Rightarrow_{lmax} (ab, abb)$. This is because rule 2 involves cell 1, hence it is not possible to apply rule 2 in parallel with rule 3 without applying at the same time rule 1. Yet again, this contradicts our hypothesis, hence it has to be $F(1) = F(2)$. The same reasoning applies to the case $F(2) \neq F(3)$: $M_0 \rightsquigarrow_{lmax} (a,1)(a,1)(b,1)(a,2)(b,2)$, but $C_0 \not\Rightarrow_{lmax} (aab, ab)$. Therefore, it has to be $F(1) = F(2) = F(3)$ but this is in contrast with our earlier observation that it has to be $F(1) \neq F(3)$ in order to have $DUE \equiv_{lmax} \mathcal{N}(DUE, F)$.

Thus, we can conclude that there is no $F : \{1,2,3\} \to \mathbb{N}$ such that $DUE \equiv_{lmax} \mathcal{N}(DUE, F)$. □

Proposition 5.2 shows that, in the case of locally-maximal parallelism, it is not always possible for the corresponding PTL-net to mimic the behavior of the original network of cells. The intuitive reason for this is that interaction rules may involve several cells at the same time and locally-maximal parallelism for NC's is defined with respect to the cells involved rather than with respect to the rules applied; localities in PTL-nets are instead associated with transitions and determine which transitions fire in parallel within a step of executions irrespectively of the places involved. However, for an NC with interaction rules of input radius at most 1 (i.e., interaction rules that involve at most one cell in their left-hand side), it is easy to construct a corresponding PTL-net which exhibits an equivalent behavior even for locally-maximally parallelism. This is the case for the basic model of P systems as shown in [13, 12].

A similar result can be obtained for minimal parallelism.

Proposition 5.3 *There exists a network of cells Π with 3 rules such that, for all $F : \{1,2,3\} \to \mathbb{N}$, we have that $\Pi \not\equiv_{min} \mathcal{N}(\Pi, F)$.*

Proof. Let $DUE = (\{a,b\}, aa, bb, R)$ be the NC used in the proof of Proposition 5.2 with its initial configuration $C_0 = (aa, bb)$. Then, let us suppose that $DUE \equiv_m \mathcal{N}(DUE, F)$ for some $F : \{1,2,3\} \to \mathbb{N}$. PTL-net $\mathcal{N}(DUE, F)$ is the same as the one defined in the proof of Proposition 5.2.

Now, we observe that if $F(1) = F(2) = F(3)$, then, by firing only transition 1 with multiplicity 1, $M_0 \rightsquigarrow_{min} (a,1)(a,1)(a,1)(b,2)(b,2)$, but $C_0 \not\Rightarrow_{min} (aaa, bb)$. This is because rule 1 involves only cell 1, hence, since we are operating according to the minimal parallelism, it is not possible to apply rule 1 with multiplicity 1 without applying in parallel at the same time at least another rule involving cell 2. This contradicts our hypothesis, hence it cannot be $F(1) = F(2) = F(3)$.

Next, if $F(1) \neq F(2)$, then, by applying rule 2 with multiplicity 1, $C_0 \Rightarrow_{min} (ab, ab)$ but $M_0 \not\rightsquigarrow_{min} (a,1)(b,1)(a,2)(b,2)$. This is because, since we are operating according to minimal parallelism, it is not possible to fire transition 2 with multiplicity 1 without firing in parallel at the same time at least another transition belonging to locality $F(1) \neq F(2)$. The same reasoning apply to case $F(2) \neq F(3)$. Therefore, it has to be $F(1) = F(2)$ and $F(2) = F(3)$ but this is in contrast with our earlier observation that it cannot be $F(1) = F(2) = F(3)$ in order to have $DUE \equiv_{min} \mathcal{N}(DUE, F)$.

Thus, we can conclude that there is no $F : \{1,2,3\} \to \mathbb{N}$ such that $DUE \equiv_{min} \mathcal{N}(DUE, F)$. □

Proposition 5.3 shows that, even for minimal parallelism, it is not always possible for the corresponding PTL-net to mimic the behavior of the original network of cells. Proposition 5.2 and Proposition 5.3 are both based on the fact that an NC may contain both rules of input radius 1 with a local scope and rules of radius greater than 1 which in a sense belong to different cells at the same time.

We pass now to consider the reverse problem of finding for every PTL-net a corresponding network of cells with an equivalent behavior.

Definition 14. *Let $N = (P, T, W, M_0, L)$ be a PTL-net.*

1. *A* cell mapping for N *is any surjective function $C : P \to \{1, \ldots, n\}$ with $n \geq 1$.*
2. *For all $t \in T$, for any cell mapping C for N, the* pre-partition of t (w.r.t. C)*, denoted by $prep_C(t)$, is the string $(w_1, c_1)(w_2, c_2) \ldots (w_k, c_k)$ such that $pre_N(t) = w_1 w_2 \ldots w_k$, for all $1 \leq i \leq k$, $w_i \neq \lambda$ and $C(\aleph(w_i)) = \{c_i\}$, and, for all $1 \leq i, j \leq k$ with $i \neq j$, $C(\aleph(w_i)) \neq C(\aleph(w_j))$.*
3. *For all $t \in T$, for any cell mapping C for N, the* post-partition of t (w.r.t. C)*, denoted by $postp_C(t)$, is the string $(w_1, c_1)(w_2, c_2) \ldots (w_k, c_k)$ such that $post_N(t) = w_1 w_2 \ldots w_k$, for all $1 \leq i \leq k$, $w_i \neq \lambda$ and $C(\aleph(w_i)) = \{c_i\}$, and, for all $1 \leq i, j \leq k$ with $i \neq j$, $C(\aleph(w_i)) \neq C(\aleph(w_j))$.*
4. *For every marking M of N, for any cell mapping C for N, the* partition of M (w.r.t. C)*, denoted by $part_C(M)$, is the string $(w_1, c_1)(w_2, c_2) \ldots (w_k, c_k)$ such that $M = w_1 w_2 \ldots w_k$, for all $1 \leq i \leq k$, $w_i \neq \lambda$ and $C(\aleph(w_i)) = \{c_i\}$, and, for all $1 \leq i, j \leq k$ with $i \neq j$, $C(\aleph(w_i)) \neq C(\aleph(w_j))$.*
5. *For every marking M of N, for any cell mapping $C : P \to \{1, \ldots, n\}$ for N, the* extended partition of M (w.r.t. C)*, denoted by $exp_C(M)$, is the tuple $(w_1, w_2, \ldots, w_n)$ such that $M = w_1 w_2 \ldots w_n$ with $part_C(M) = (w_{c_1}, c_1)(w_{c_2}, c_2) \ldots (w_{c_k}, c_k)$ for some $\{c_1, c_2, \ldots, c_k\} \subseteq \{1, 2 \ldots, n\}$ and $w_i = \lambda$ for all $i \notin \{c_1, c_2, \ldots, c_k\}$.*

Thus, given a network of cells N, we use the cell mapping to associate places to certain cells and we use the concept of partition to distribute the places (i.e., their multiple occurrences) to these cells. Specifically, we give the following definition.

Definition 15 (corresponding NC). *Let $N = (P, T, W, M_0, L)$ be a PTL-net and let $C : P \to \{1, \ldots, n\}$ be a cell mapping for N. The* network of

cells corresponding to N (w.r.t. C) *is* $\mathcal{P}(N,C) = (P, w_1, w_2, \ldots, w_n, R)$ *where:* $exp_C(M_0) = (w_1, w_2, \ldots, w_n)$ *and*

$$R = \{(u_1,i_1)\ldots(u_k,i_k) \to (v_1,j_1)\ldots(v_h,j_h) \mid \\ t \in T, prep_C(t) = (u_1,i_1)\ldots(u_k,i_k), postp_C(t) = (v_1,j_1)\ldots(v_h,j_h)\}.$$

Thus, we can construct different NC's corresponding to the same PTL-net depending on the way we decide to assign places to cells; the unique constraint is that multiple occurrences of the same place have to remain confined within the same cell. At one end, a PTL-net can be translated into an NC with one cell where all the rules have input radius equal 1; every rule corresponds to a transition and these rules are no more than multiset rewriting rules. At the opposite end, a PTL-net can be translated into an NC with as many cells as its places; every rule corresponds to a transition and the its input radius is equal to the cardinality of the support of the pre-multiset of this transition. Notice that the locality mapping of a PT-net has no direct counterpart in the corresponding NC's.

Next, we introduce the notion of equivalence between a PTL-net and its corresponding NC's.

Definition 16 (m-equivalence). *Let* $N = (P, T, W, M_0, L)$ *be a PTL-net and let* $C : P \to \{1, \ldots, n\}$ *be a cell mapping for* N. *For* $\mathrm{m} \in \{free, seq, max, lmax, min\}$, *we say that* N *is* m-equivalent to $\mathcal{P}(N,C)$, *and we write* $N \approx_{\mathrm{m}} \mathcal{P}(N,C)$ *if, for every two markings* M_1, M_2, $M_1 \rightsquigarrow_{\mathrm{m}} M_2$ *iff* $exp_C(M_1) \Rightarrow_{\mathrm{m}} exp_C M_2$.

Thus, we can state the fundamental property which relates PTL-nets and corresponding NC's.

Proposition 5.4 *For any PTL-net* $N = (P, T, W, M_0, L)$, *for any cell mapping* C *for* N, *for* $\mathrm{m} \in \{free, seq, max\}$, *we have that* $N \approx_{\mathrm{m}} \mathcal{P}(N,C)$. *Moreover, if* $L(T) = \{l\}$ *for some* $l \geq 0$, *then, for any cell mapping* C *for* N, *for* $\mathrm{m} \in \{lmax, min\}$, *we have that* $N \approx_{\mathrm{m}} \mathcal{P}(N,C)$.

Therefore, the general equivalence between a PTL-net and its corresponding NC's can only be established for free-parallelism, sequential execution and maximal parallelism. For locally-maximal parallelism and minimal parallelism, the aforementioned equivalence can be established only for PTL-nets where all transitions are assigned to the same locality. In fact, the following results holds.

Proposition 5.5 *There exists a PTL-net* N *such that, for any cell mapping* C *for* N, *for* $\mathrm{m} \in \{lmax, min\}$, *we have that* $N \not\approx_{\mathrm{m}} \mathcal{P}(N,C)$.

Proof. Let $NONE$ be the PTL-net of Figure 2 where: transition R is assigned locality 1, transition S is assigned locality 2, and transition T is assigned locality 2; the initial marking of $NONE$ is $M_0 = aabb$. Then, let us suppose that there is a cell mapping C for $NONE$ such that $NONE \approx_{\mathrm{m}} \mathcal{P}(NONE, C)$.

Now, if $C(\{a,b\} = \{1\}$ (i.e., if $\mathcal{P}(NONE, C)$ contains only one cell), then it is obvious that $NONE \not\approx_{\mathrm{m}} \mathcal{P}(NONE, C)$. Therefore, it has to be $C(\{a,b\} = \{1,2\}$

(i.e., $\mathcal{P}(NONE, C)$ has to contain two cells) in order to be able to distinguish between locality 1 and 2.

Then, if $C(a) = 1$ and $C(b) = 2$, then, by Definition 15, $\mathcal{P}(NONE, C)$ contains rules: $(a,1) \to (aa,1)$, $(a,1)(b,2) \to (b,2)$, and $(b,2) \to (bb,2)$. Thus, $M_0 \rightsquigarrow_{lmax} aabb$ by firing transition R and transition S belonging to the same locality, but $exp_C(M_0) = (aa, bb) \not\Rightarrow_{lmax} (aa, bb)$. This is because rule $(a,1)(b,2) \to (b,2)$ involve cell 1 as well as cell 2, hence we cannot apply rule $(a,1) \to (aa,1)$ in parallel with rule $(a,1)(b,2) \to (b,2)$ without applying at the same time rule $(b,2) \to (bb,2)$. This contradicts our hypothesis. For symmetry, the same reasoning apply to the case $C(a) = 2$ and $C(b) = 1$.

Therefore, we can conclude that, for any cell mapping C for N, we have that $N \not\approx_{lma}$

For ıe cases as abov $\mathcal{N} \not\approx_{min}$ $\mathcal{P}(N, C$

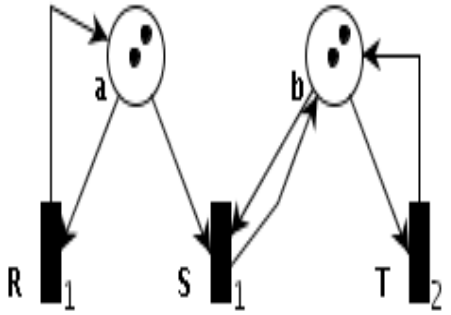

Fig. 2. PTL-net $NONE$ with its initial marking where transition R is assigned to locality 1, transition S is assigned to locality 2, and transition T is assigned to locality 2.

Finally, we stress once more the differences between cells and places. Cells of an NC are bags of objects that can contain multiple occurrences of different symbols, each one of them representing a different sort of objects; cells are seen as distinct components of a larger system whose behavior is given by their interactions; the adopted semantics determines which and how many cells become active from time to time. Places of a PTL-net store a number of tokens, each one of them representing a distinct occurrence of a specific place; places with their number of tokens represent resources that have to be acquired by transitions in order to fire, and, in PTL-nets, one usually abstracts from the actual location of these resources. Therefore, when translating a PTL-net into an NC, one has to make some extra assumptions about the assignment of places to cells, although, in general, one can

always see a PTL-net as an NC with one cell containing a finite set of multisets of rewriting rules.

6 Case-Studies

We present NC's solutions to two classical synchronization problems: producer/consumer and dining philosophers. The proposed models are derived from standard solutions based on Petri Nets.

6.1 Producer/Consumer

We consider the simpler version of the producer/consumer system from [20]: distinguished items are produced, delivered to a buffer, later removed from the buffer, and finally consumed. The buffer is assumed to have capacity for one item. Moreover, like in [20], we abstract from any concrete instance of the aforementioned operations and we focus only on the "interplay" between produce/deliver and remove/consume, and on the events necessary for their synchronization.

Specifically, we consider two sub-systems named *producer* and *consumer*, respectively, which "synchronize" (or "interact", or "communicate") through a shared buffer. The producer has two states: "ready to produce" and "ready to deliver". The consumer has two states: "ready to remove" and "ready to consume". The buffer has two states: "filled" and "empty". In state "ready to produce", the producer executes the operation "produce" and moves to state "ready to deliver"; in state "ready to produce", if the buffer is "empty", the producer executes the operation "deliver", which fills the buffer, and moves back to state "ready to produce". Similarly, in state "ready to remove", if the buffer is "empty", the consumer execute the operation "remove", which empties the buffer, and moves to state "ready to consume"; in state "ready to consume", the consumer executes the operation "consume" and moves back to state "ready to remove".

PTL-Net Representation

The PTL-net model of the aforementioned producer/consumer system, which is presented in [20], is reported in Figure 3, with its initial marking AEG, where:

$A \equiv$ "ready to produce",
$B \equiv$ "ready to deliver",
$F \equiv$ "filled",
$E \equiv$ "empty",
$G \equiv$ "ready to remove",
$H \equiv$ "ready to consume",
$p \equiv$ "produce",
$d \equiv$ "deliver",

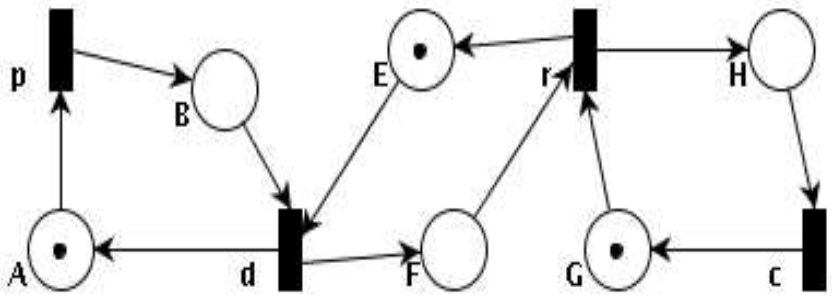

Fig. 3. PTL-net model of a producer/consumer system.

$r \equiv$ "remove",
$c \equiv$ "consume".

The unique transition enabled at the initial marking AEG is transition p ("produce"), and, when fired, it transfers a token from place A ("ready to produce") to place B ("ready to deliver"). Next, transition d ("deliver") is enabled and, when fired, it produces a token in place F (i.e., the buffer is filled) and a token in place A ("ready to produce"). Then, transition r ("remove") can fire which produces a token in place E (i.e., the buffer is emptied) and a token in place H ("ready to consume"). Finally, a token can be returned to place G ("ready to remove") by firing transition c ("consume").

We remark that:

- seq-execution and free-execution are non-deterministic; after transition d has fired, both transition p and transition r are enabled and, after transition r has fired, both transition d and transition c are enabled;
- max-execution is deterministic;
- if transitions p and d are assigned to a certain locality (i.e., the producer) and transition r and c to another one (i.e., the consumer), then lmax-execution is equivalent to free-execution;
- if transitions p and d are assigned to a certain locality (i.e., the producer) and transition r and c to another one (i.e., the consumer), then min-execution is equivalent to max-execution;
- irrespectively of the semantics, in each step, at least one transition is always enabled (*liveness*);
- irrespectively of the semantics, the consumer can consume only after having received an item from the buffer (i.e., only after having performed a "remove" operation);
- irrespectively of the semantics, after having produced an item, the producer has to deliver it into the buffer before returning to producer; the delivering can take place only when the buffer has been emptied.

NC Representation

As pointed out in Section 5, it is possible to find different NC's corresponding to the PTL-net of Figure 3. Here, in order to model the aforementioned producer/consumer system, we consider an NC PC with 3 cells. Cell 1 represents the producer, cell 2 represents the consumer, whereas cell 3 represents the buffer.

Cell 1 stores an object which specifies the states of the producer; this can be either A ("ready to produce") or B ("ready to deliver"). Similarly, cell 2 stores an object which specifies the state of the consumer; this can be either G ("ready to remove") or H ("ready to consume"). Cell 3 instead stores an object representing the state of the buffer, either F ("filled") or E ("empty"). The initial configuration is the tuple (A, G, E).

The desired behavior is then implemented by considering the following rules:

1. $(A, 1) \rightarrow (B, 1)$,
2. $(B, 1)(E, 2) \rightarrow (A, 1)(F, 3)$,
3. $(G, 2)(F, 3) \rightarrow (H, 2)(E, 3)$,
4. $(H, 2) \rightarrow (G, 2)$.

Yet again, we have that:

- the application of the rules is sequential and freely-parallel;
- the maximally-parallelism is deterministic;
- irrespectively of the semantics, in each step, at least one rule is always applicable;
- irrespectively of the semantics, the consumer can consume only after having received an item from the buffer;
- irrespectively of the semantics, after having produced an item, the producer has to deliver it into the buffer before returning to producer; the delivering can take place only when the buffer has been emptied.

Also, the PT-net $\mathcal{N}(PC)$ is the same as the PTL-net of Figure 3 except for the naming of places and transitions (i.e., they define two isomorphic graphs). In fact, rule 1 corresponds to transition p in the PT-net of Figure 3, rule 2 corresponds to transition d in the PT-net of Figure 3, rule 3 corresponds to transition r in the PT-net of Figure 3, and rule 4 corresponds to transition c in the PT-net of Figure 3. Moreover, for $F : \{1, 2, 3, 4\} \rightarrow \mathbb{N}$ such that $F(1) = F(2)$, $F(3) = F(4)$ and $F(1) \neq F(3)$, we have that $PC \equiv_{lmax} \mathcal{N}(PC, F)$ and $PC \equiv_{min} \mathcal{N}(PC, F)$. The vice versa is also true: for the PTL-net of Figure 3, if transitions p and d are assigned to a certain locality (i.e., the producer) and transition r and c to another one (i.e., the consumer), then, for m $\in \{free, seq, max, lmax, min\}$, PTL-net of Figure 3 is m-equivalent to PC. Therefore, for this version of the producer/consumer system, there is a sort of direct transcription of the PTL-net model into an NC model whose rules are able to represent exactly the same type of interactions between a producer and a consumer.

Remark 4. For the present version of producer/consumer, the fundamental properties are: there is always a transition enabled (*liveness*), the producer always alternates between "ready to consume" and "ready to deliver", and the consumer always alternates between "ready to remove" and "ready to consume". These properties are formally proved in [20] for the PTL-net of Figure 3 and these results can be directly transferred to our NC model; at some extent, this gives a flavor of the sort of properties which could be proved for NC's (or any other membrane systems) by using techniques developed in the area of Petri nets.

Next, we describe an evolution-communication model of the producer/consumer systems that is not a direct translation of the PTL-net solution from [20] but is inspired by the idea of P systems as systems where transformations involve only objects inside one specific cell and communication is responsible for moving objects from one cell to the other. In other words, we consider an NC with interaction rules of input radius 1 which does not rely on the simultaneous synchronization of two different cells.

Similarly to what was done before, we consider an NC $PC1$ with 3 cells: cell 1, the producer, cell 2, the consumer, and cell 3, the buffer. The initial configuration of $PC1$ is the tuple (A, G, E).

Cell 1 always moves from state A ("ready to produce") to B ("ready to deliver") by replacing in its inside object A with object B. In state B, after having received an object E from cell 3 (i.e., after having been notified that the buffer is empty), cell 3 produce in its inside an object A and object F; object F is then sent to cell 3 to fill the buffer. In state G ("ready to remove"), cell 2 always waits to receive an object F from cell 3 in order produce in its inside an object H (representing state "ready to consume") and an object E; object E is then sent to cell 3 to notify that the buffer is now empty. This behavior is implemented by means of the following rules:

1. $(A, 1) \to (B, 1)$,
2. $(E, 3) \to (E, 1)$,
3. $(BE, 2) \to (AF, 1)$,
4. $(F, 1) \to (F, 3)$,
5. $(F, 3) \to (F, 2)$,
6. $(GF, 2) \to (HE, 2)$,
7. $(H, 2) \to (G, 2)$,
8. $(E, 2) \to (E, 3)$.

Thus, all these rule have input radius equal to 1 and interactions between cell 1 (cell 2) are achieved through an effective exchange of objects. However, irrespectively of the semantics adopted, the aforementioned fundamental properties can still be observed: at any time, at least one rule is applicable, the producer always alternates between "ready to consume" and "ready to deliver", and the consumer always alternates between "ready to remove" and "ready to consume". Moreover, the producer can deliver only when the buffer is empty, and the consumer can

remove only when the buffer is filled. Also, notice that $PC1$ is somehow "more concurrent" than PC: there is always some communication which can be executed in parallel with the internal change of state. Nevertheless, maximal parallelism still leads to a deterministic behavior for $PC1$ and the movement of objects E and F does not affect the behavior of producer and consumer.

PTL-net $\mathcal{N}(PC1)$ is given in Figure 4.

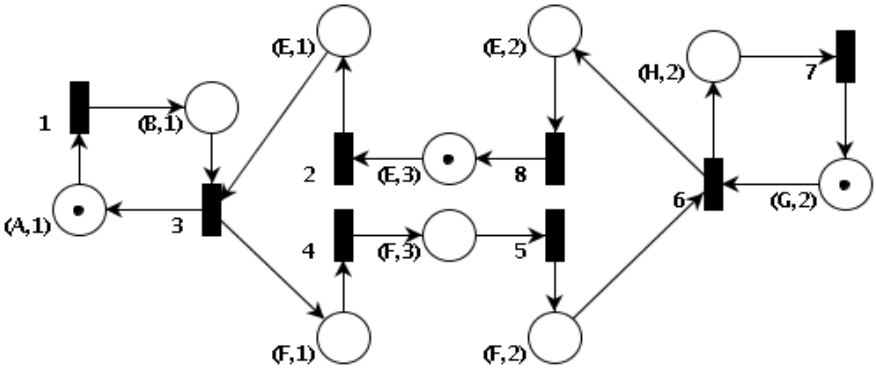

Fig. 4. PTL-net $\mathcal{N}(PC1)$ with its initial marking.

Remark 5. The solution based on the NC PC uses rules of input radius at most 2 and cooperation degree at most 1, whereas the solution based on the $PC1$ uses rules of input radius at most 1 and cooperation degree at most 2. In some sense, this shows a trade-off between the cooperation at level of cells and the cooperation at the level of objects.

Remark 6. If we do not consider the names of places and transitions, the PTL-net of Figure 4 is the same as the PTL-net of Figure 3 excepts for the presence of some "unary" transitions which correspond to the movement of objects E and F between cell 1, cell 2 and cell 3. Indeed, by using standard structural methods of Petri nets (e.g., see [11]), the aforementioned transitions could be removed so to have a PTL-net which is the same as the one of Figure 3 and inherits the same structural properties proved in [20]. Thus, the translation into PTL-net allows us to relate two different models of NC's in terms of structural properties which are somehow hidden in the two different types of rules used.

7 Dining Philosophers

We consider a distributed system where each resource is shared by two components, and each subsystem simultaneously requires two resources in order to start its

activities. The problem of modeling such a resource sharing scenario is classically formulated as the problem of the dining philosophers: five philosophers are sitting around a table, each philosopher has his own plate and can be eating or thinking (i.e., not eating). In order to eat, a philosopher needs two forks, but there are only two forks next to each plate so that no two neighboring philosophers may be eating simultaneously.

PTL-net representation

The five philosophers are denoted by A, B, C, D and E; the five forks are denoted by f_0, f_1, f_2, f_3 and f_4. For each $X \in \{A, B, C, D, E\}$, we denote by $l(X)$ its left fork and by $r(X)$ its right fork. Specifically, we set: $l(A) = f_0$, $r(A) = f_1$, $l(B) = f_1$, $r(B) = f_2$, $l(C) = f_2$, $r(C) = f_3$, $l(D) = f_3$, $r(D) = f_4$, $l(E) = f_4$ and $r(E) = f_0$ (i.e. A sits between E and B, B sits between A and C, C sits between B and D, D sits between C and E, and E sits between D and A).

Thus, a PTL-net FDP is constructed that contains: for each $X \in \{A, B, C, D, E\}$, a place X_t ("X_t is thinking") and a place X_e ("X_e is eating"), for each $0 \leq i \leq 4$, a place f_i ("f_i is available"), for each $X \in \{A, B, C, D, E\}$, a transition X_p ("X picks up forks") with $pre_{FDP}(X_p) = l(X)X_t r(X)$ and $post_{FDP}(X_p) = X_e$, and a transition X_r ("X returns forks") with $pre_{FDP}(X_r) = X_e$ and $post_{FDP}(X_r) = l(X)X_t r(X)$. The initial marking of FDP is $A_t B_t C_t D_t E_t f_0 f_1 f_2 f_3 f_4$ (i.e., all philosophers are thinking and all forks are available).

Figure 5 depicts the sub-net of an FDP modeling a philosopher. This classical PTL-net model of the dining philosophers is taken from [20].

At the initial marking $A_t B_t C_t D_t E_t f_0 f_1 f_2 f_3 f_4$, for $X \in \{A, B, C, D, E\}$, each transition X_p ("X picks up forks") is enabled by giving all philosophers a chance to start eating. However, irrespectively of the execution mode adopted, no two neighboring philosophers can start eating at the same time because they share one fork and, for all $0 \leq i \leq 4$, f_i contains only one token. Thus, in the first step, depending on the execution mode, a certain number of philosophers which are not neighbors start eating by firing at least one transition X_p, whereas the other philosophers keep thinking. Then, depending on the execution mode, some other philosophers may start eating, whereas the eating philosophers may release their forks and return thinking, and so on.

The fundamental property of the PTL-net FDP is that it avoids the deadlocks: FDP never produces a marking which no transition is enabled at. Moreover, from time to time, FDP offers every philosopher the chance to start eating (i.e., it avoids scenarios where no philosopher can ever start eating), although fairness is not guaranteed (i.e., there may be some philosophers which always alternate between thinking and eating with the other philosophers thinking indefinitely). Yet again, this structural properties are formally proven in [20].

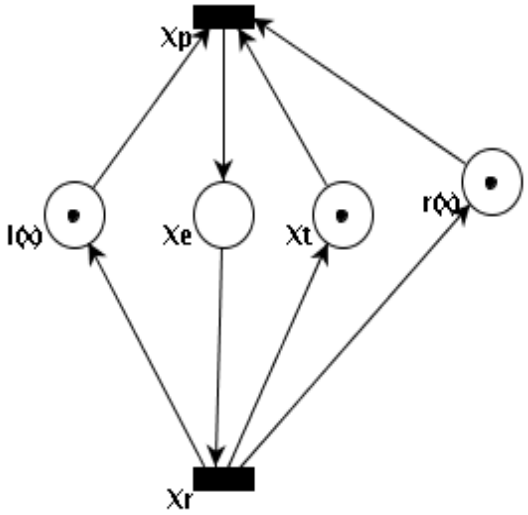

Fig. 5. PTL-net representation of a dining philosopher with its initial marking.

NC Representation

By starting from the PTL-net FDP, we construct an NC PH with 6 cells: cell 1 represents philosopher A, cell 2 represents philosopher B, cell 3 represents philosopher C, cell 4 represents philosopher D, cell 5 represents philosopher E, and cell 6 representing the table. If we denote by $ph(i)$ the philosopher represented by cell i, for all $1 \leq i \leq 5$, then cell i always contain either an object $ph(i)_t$ ("$ph(i)$ is thinking") or an object $ph(i)_e$ ("$ph(i)$ is eating"). Also, as in the previous PTL-net representation, for each $X \in \{A, B, C, D, E\}$, we denote by $l(X)$ its left fork and by $r(X)$ its right fork. Cell 6, the table, always contain the currently available forks: the availability of a fork f_i, for all $0 \leq i \leq 4$, is represented as occurrence of an object f_i inside cell 6. The initial configuration of PH is the tuple $(A_t, B_t, C_t, D_t, E_t, f_0 f_1 f_2 f_3 f_4)$.

The behavior of the five dining philosophers is then implemented by means of a set of rules which, for all $1 \leq i \leq 5$, contains the rules:

1. $(ph(i)_t, i)(l(ph(i))\, r(ph(i)), 6) \rightarrow (ph(i)_e, i)$,
2. $(ph(i)_e, i) \rightarrow (ph(i)_t, i)(l(ph(i))\, r(ph(i)), 6)$.

Specifically, for all $1 \leq i \leq 5$, rule 1 models transition $ph(i)_p$ ("$ph(i)$ picks up forks") of the PTL-net FDP, whereas rule 2 models transition $ph(i)_r$ ("$ph(i)$ returns forks) of the PTL-net FDP. In fact, the NC PH is the PTL-net $\mathcal{N}(FDP, C)$ with $C(A_t) = C(A_e) = 1$, $C(B_t) = C(B_e) = 2$, $C(C_t) = C(C_e) = 3$,

$C(D_t) = C(D_e) = 4$, $C(E_t) = C(E_e) = 6$, and $C(f_i) = 6$ for all $0 \leq i \leq 4$. Therefore, even for the NC PH, we can say that deadlock is avoided.

Another NC representation of the five dining philosophers can be obtained by distributing the forks to five different cells. Specifically this means considering an NC $PH1$ obtained from PH by removing cell 6 and adding a cell $6+i$, for all $0 \leq i \leq 4$; each cell $6+i$ contains an occurrence of object f_i whenever fork f_i is available. The initial configuration of $PH1$ is the tuple $(A_t, B_t, C_t, D_t, E_t, f_0, f_1, f_2, f_3, f_4,)$.

Then, for all $1 \leq i \leq n$, the new set of rules contains the rules:

1. $(ph(i)_t, i)(l(ph(i)), c(l(ph(i))))(r(ph(i)), c(r(ph(i)))) \rightarrow (ph(i)_e, i)$,
2. $(ph(i)_e, i) \rightarrow (l(ph(i)), c(l(ph(i))))(r(ph(i)), c(r(ph(i))))$.

where, for all $1 \leq i \leq 5$, $c(l(ph(i)))$ and $c(r(ph(i)))$ denote the respective locations of these two forks.

The NC $PH1$ contains rules of input radius at most 3 and cooperation degree at most 1, and it corresponds to the PTL-net $\mathcal{N}(FDP, C)$ with $C(A_t) = C(A_e) = 1$, $C(B_t) = C(B_e) = 2$, $C(C_t) = C(C_e) = 3$, $C(D_t) = C(D_e) = 4$, $C(E_t) = C(E_e) = 6$, and $C(f_i) = 6 + i$ for all $0 \leq i \leq 4$. Therefore, with respect to their PTL-net representation, the NC PH and NC $PH1$ cannot be distinguished.

8 Discussion

Networks of cell (NC's, for short) are a general class of P systems which encompass the essential features of evolution/communication of both P systems and tissue P systems. Rules in NC's allow different cells to synchronize in order to consume some multisets from their inside and produce some new multisets inside some other cells. As we have seen, this means that we can express within the framework of NC's both forms of coupled transports, like boundary rules and standard evolution rules with communication controlled by targets $here, in, out$. However, the structure of an NC does not necessarily corresponds to a graph or a tree; NC's of cells abstract from the underlying structure by focusing only on the interactions which can take place between the cells present in the system. In fact, these cells can be equally thought as being physically connected in some way which makes possible for the interactions to take place, or as randomly bumping into each other and interacting whenever it is possible. In a sense, such an approach is closer to the idea of a Petri net as a collection of transitions which can fire when certain resources become available, with some of these transitions competing for the same set of resources. The difference is that in NC's resources are objects which are specifically placed inside certain cells. This has led us to two important observations:

- Despite being able to translate every NC into a PTL-net by using a construction analogous to the one used in [7, 13, 12], in the case of locally-maximal parallelism and minimal parallelism, it is not always possible for the corresponding PTL-net to mimic the behavior of the original network of cells; locally-maximal

parallelism for PTL-nets is defined with respect to the localities which are assigned to the transitions; these localities then determine which transitions can possibly fire in parallel at the same time irrespectively of the places involved; rules of NC's involving more than one cells are instead not assigned a priori to any cell, and locally-maximally parallelism and minimal parallelism are defined with respect to a cell that can get involved in some interactions from time to time depending on a particular distribution of objects;

- For a given PTL-net, it is possible to find different corresponding NC's depending on the way places are assigned to the cells; the only restriction is that multiple occurrences of the same place has to remain confined within the same cell; for instance, for the five dining philosophers, it has been possible to produce two NC-based models: one with 6 cells and rules of input radius 2, and another one with 10 cells and rules of input radius 3.

In other words, in P systems, one naturally reasons about components (e.g., producer, consumer, buffer) and these are usually seen as being separate cells (or membranes). Also, one naturally distinguishes between operations affecting the inner state of a membrane and the operations involving interactions between different membranes. Moreover, in membrane systems, the inner state of a membrane can be given by an arbitrarily large multiset; this allows us to combine cooperation at the level of objects with interaction between different cells. Petri nets, with their graphical notation, are centered around the idea of resources which have to be acquired (tokens inside places) before certain actions can be taken; this facilitates the reasoning about properties like causality (the execution of certain actions depends on the execution of some others), concurrency (certain group of action can always be executed in parallel), and conflicts (certain actions compete with some others for the usage of certain resources); in membrane systems, these structural properties instead remains somehow hidden in the formalism used for representing the rules.

The present research can be continued in several directions. For instance, from a computational point of view, although we know that NC's with a maximally-parallel semantics are computationally complete, the computational power of NC's of cells with other semantics deserves further investigations especially for what concerns the size and types of rules used. Moreover, as pointed out in [24], interaction rules of NC's can express forms of cooperative communication other than symport/antiport or conditional uniport, and the computational power of these forms of communication is not yet fully understood. Then, for what concerns the relationships between P systems and Petri nets, as observed in [12], other features of P systems (without being limited to NC's) may or may not be representable in Petri nets. However, for some classes of P systems, their Petri-net representation offers the possibility of analyzing their behavior with respect to certain structural properties which can be formally proved for Petri nets and which are thoroughly managed through a plethora of tools [2]. Finally, we remark that, although one may see Petri nets as a low-level implementation of P systems, there are classes of high-level Petri Nets (e.g., colored Petri nets [11], objects nets [11], system nets

[20]) with features usually not considered for P systems, such as: types, variables, and predicates. The need for these features in P systems may be checked for specific modeling purposes against appropriate case-studies.

Acknowledgements

The research of Francesco Bernardini is supported by NWO, Organisation for Scientific Research of The Netherlands, project 635.100.006 "VIEWS".

References

1. The P systems web page. `http://psystems.disco.unimib.it`.
2. Petri nets world - Petri nets tools database. `http://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html`.
3. F. Bernardini and M. Gheorghe. Cell Communication in Tissue P Systems: Universality Results. *Soft Computing*, 9, 9 (2005), 640–649.
4. F. Bernardini and V. Manca. P Systems with Boundary Rules. In Gh. Păun, G. Rozenberg, A. Salomaa, and C. Zandron, editors, *Membrane Computing. International Workshop, WMC-CdeA 02, Curtea de Argeş, Romania, August 19-23, 2002. Revised Papers*. Volume 2597 of *Lecture Notes in Computer Science*, Springer, 2003, 107–118.
5. F. Bernardini, F.J. Romero-Campero, M. Gheorghe, and M.J. Pérez-Jiménez. A Modelling Approach Based on P Systems with Bounded Parallelism. In H.J. Hoogeboom, Gh. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing, Seventh International Workshop, WMC7, Leiden, The Netherlands, July 2006*. Volume 4361 of *Lecture Notes in Computer Science*, Springer, 2007, 49–65.
6. G. Ciobanu, L. Pan, Gh. Păun, M.J. Pérez-Jiménez. P systems with minimal parallelism. *Theoretical Computer Sci.*, to appear.
7. S. Dal Zilio and E. Formenti. On the Dynamics of PB Systems: A Petri Net View. In C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July, 17-22, 2003, Revised Papers*. Volume 2933 of *Lecture Notes in Computer Science*, Springer, 2004, 153–167.
8. J. Desel, W. Reisig, and G. Rozenberg, editors. *Lectures on Petri Nets and Concurrency*. Volume 3098 of *Lecture Notes in Computer Science*. Springer, 2004.
9. R. Freund. Asynchronous P Systems and P Systems Working in the Sequential Mode. In G. Mauri, Gh. Păun, M. J. Pérez-Jiménez, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing. International Workshop, WMC 2004, Milan, Italy, June 2004. Revised and Invited Papers*. Volume 3365 of *Lecture Notes in Computer Science*, Springer, 2005, 36–62.
10. R. Freund, L. Kari, M. Oswald, and P. Sosík. Computationally Universal P Systems without Priorities: Two Catalysts are Sufficient. *Theoretical Computer Science*, 330 2 (2005), 251–266.
11. C. Girault and R. Valk. *Petri Nets for Systems Engineering*. Springer, 2003.
12. J. Kleijn and M. Koutny. Synchrony and Asynchrony in Membrane Systems. In H.J. Hoogeboom, Gh. Paun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing, Seventh International Workshop, WMC7, Leiden, The Netherlands, July 2006*. Volume 4361 of *Lecture Notes in Computer Science*, Springer, 2007, 66–85.

13. J. Kleijn, M. Koutny, and G. Rozenberg. Towards a Petri Net Semantics for Membrane Systems. In R. Freund, Gh. Păun, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing: 6th International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised, Selected and Invited Papers*. Volume 3850 of *Lecture Notes in Computer Science*, Springer, 2006, 292–309.
14. S.N. Krishna and A. Păun. Results on Catalytic and Evolution-Communication P Systems. *New Generation Computing*, 22, 4 (2004), 377–394.
15. C. Martin-Vide, Gh. Păun, J. Pazoz, and A. Rodriguez-Paton. Tissue P Systems. *Theoretical Computer Sci.*, 296, 2 (2003), 295–326
16. C. Martín-Vide, Gh. Păun, and G. Rozenberg. Membrane Systems with Carriers. *Theoretical Computer Science*, 270, 1-2 (2002), 779–796.
17. Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
18. Gh. Păun. *Membrane Computing. An Introduction*. Springer, Berlin, 2002.
19. Gh. Păun, Y. Sakakibara, and T. Yokomori. Membrane Systems on Graphs of Restricted Forms. *Publicationes Mathematicae Debrecen*, 60 (2002), 635–660.
20. W. Reisig. *Elements of Distributed Algorithms. Modelling and Analysis with Petri Nets*. Springer, 1998.
21. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets I: Basic Models*. Volume 1491 of *Lecture Notes in Computer Science*. Springer, 1998.
22. W. Reisig and G. Rozenberg, editors. *Lectures on Petri Nets II: Applications*. Volume 1492 of *Lecture Notes in Computer Science*. Springer, 1998.
23. G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages*, volume 1–3. Springer, 1997.
24. S. Verlan, F. Bernardini, M. Gheorghe, and M. Margenstern. Generalized Communicating P Systems. Submitted, 2007.
25. S. Verlan, F. Bernardini, M. Gheorghe, and M. Margenstern. Computational Completeness of Tissue P Systems with Conditional Uniport. In H.J. Hoogeboom, Gh. Păun, G. Rozenberg, and A. Salomaa, editors. *Membrane Computing, Seventh International Workshop, WMC7, Leiden, The Netherlands, July 2006*. Volume 4361 of *Lecture Notes in Computer Science*, Springer, 2007, 521–535.