MEMBRANE DIFFERENTIAL PETRI NETS FOR PERFORMANCE MODELLING OF HYBRID P-SYSTEMS

Emilian Guţuleac[1], Mihai L. Mocanu[2], Iurie Ţurcanu[1]

[1] *Computer Science Department, Technical University of Moldova, Bvd. Ştefan cel Mare nr. 168, Chişinău, Republic of Moldova, E-mail: egutuleac@mail.utm.md*

[2] *Department of Software Engineering, Faculty of Automation, Computers and Electronics, Bvd. Decebal nr. 107, University of Craiova, ROMANIA, E-mail: mocanu@software.ucv.ro*

Abstract: *P-systems, also referred to as membrane systems, are a class of computing models inspired from the structure and functioning of living cells. Efforts have been made to simulate P-systems with Petri nets, most oriented to verify their behavioral properties. In this paper, we introduce a new methodology based on dynamic rewriting Generalized Differential Petri Nets (GDPN) for performance modeling of P-systems. We define the concept of hybrid P-systems with active membranes and give its formalization based on the new approach - the Timed Membrane GDPN (also called DMH-net). An example is presented to illustrate our approach.*

Keywords: *P-systems, reconfigurable and mobile hybrid systems, Petri Nets, performance modelling.*

1. INTRODUCTION

Hybrid systems are dynamic systems exhibiting both continuous and discrete dynamic behavior. In general, a hybrid system can be described by a few pieces of information. The *state* of the system consists of vector signals, which can change according to dynamic laws in the system *data* that includes: a *flow equation* describing the continuous dynamics, a *flow set* in which flow is permitted, a *jump equation* describing the discrete dynamics, and a *jump set* in which discrete state evolution is permitted.

Initially, the term "hybrid system" denoted systems combining electrical and mechanical drivelines, and employing neural nets and fuzzy logic for modeling. But the introduction of this new category in modeling quickly brought up the benefit of having a larger class of systems within a dynamic structure that allows for far more flexibility. Thus, their area of application has been extended to many other fields. Recent observations on living organisms led to the development of a class of hybrid systems called P-systems, also referred as membrane systems.

Generally, discrete-continuous modeling and simulation is concerned with the description, analysis and optimization of the dynamic behavior encompass of hybrid systems. Among the formalisms that are used for modeling, the hybrid Petri nets (HPN) (Alla and David, 1998) and Differential Petri Nets (DPN) (Demongodin and Koussoulas, 1998) are very popular. For the simulation of P-systems, Petri nets (Qi and Mao, 2003) have been employed mostly to aid in verifying the many useful behavioral properties such as *reachability*, *boundedness*, *liveness*, etc.

We propose in this paper a new approach to express the components of continuous variant of P-systems through components of dynamic rewriting GDPN described in (Guțuleac, *et al.*, 2006), using the descriptive expressions (DE) introduced in (Guțuleac and Mocanu, 2005). Generally, in rewriting P-systems Chmsky rules (Martin-Vide and Păun, 2000) or contextual rules (Păun, 1997) have been employed for processing string objects. Our approach is different and based on the original concepts mentioned above. To make design issues and analysis procedures more transparent, we tried to deviate as little as possible from the concepts and the nets of timed HPN and DPN. Thus, we created our extension of DPN, which we call Generalized DPN (GDPN), and that is able to represent the behavior of hybrid systems in a common model. The features of GDPN accept the *negative-continuous place capacity, negative real values for continuous place marking and negative token-dependent arc cardinalities* of continuous part that permit to generalize the concept of timed HPN and DPN (Guțuleac, *et al.*, 2006). Also, in order to capture the localities and the behavior of reconfigurable and mobile hybrid systems, we introduce hybrid P-systems with active membranes and present a formalization of this concept, the Timed Descriptive Membrane *GDPN*, based on the new approach. Also called *DMH-nets*, these are nets that can modify in run-time their own structure by rewriting some rules of their descriptive expressions components.

2. P-SYSTEMS AND MEMBRANE COMPUTING

The theory of P-systems, initiated by Păun about a decade ago and described in (Păun, 2002) is developing "on top" of a class of parallel and distributed computing models, able to reproduce the structure and functioning of living cells.
Membrane computing, the field that covers the study of P-systems, is based on observations on processes taking place in the complex structure of a living cell – their evolution can be regarded as a computation driven by chemical reaction. Hybrid models on P-systems are allowed by the switch-like character of control at the molecular level (similar to genetic regulatory networks) for the development of living organisms.

The basic components of P-system are a membrane structure consisting of several cell-like membranes placed inside a main (outer most) membrane called *skin*. Graphically, a membrane structure is represented by a Venn diagram without intersection and with a unique superset. If a membrane does not contain any other membrane, it is called *elementary*. The membranes delimit *regions* (spaces between a membrane and all directly inner membranes - if any inner membrane exists) where *objects* are placed. Object symbols can be rewritten in a non-deterministic or maximally parallel manner, parallel operations can be performed on all available objects in a membrane system, and the objects can also be communicated from one region to another. In this manner, the configuration of the system can change, starting from an initial configuration and using the evolution rules; thus, we get a *computation*.

Since the P-systems were firstly introduced, a number of classes of P-systems have been proposed and investigated. Efforts have been made in the last few years in many directions such as the description, analysis and optimization of the dynamic behavior or the formal verification of the dynamical properties of these systems. These efforts rely on the use of discrete abstractions and models checking, and have been designed to cope with aspects one often encounters when analyzing these systems, like large uncertainties on the parameter values.

Some similarities between the Petri net theory and P-systems which concern the approaches to processes generated by concurrent systems may provide a form of "transfer" of results from Petri net theory into P-systems theory. The next sections of the paper will review some of our previous results on Petri nets previously applied to other classes of systems, with a potential to facilitate an algebraic characterization of some P systems. Descriptive composition operations and expressions will allow for the creation of reconfigurable *GDPN* models for P-systems.

3. LABELED GDPN

Let L be a set of labels $L = L_P \cup L_T$, $L_P \cap L_T = \varnothing$. Each place p_i labeled $l(p_i) \in L_P$ has a local state and every transition t_j is action labeled as $l(t_j) \in L_T$.

Definition 1: A labeled GDPN is a 11-tuple
$H\Gamma = < P, T, Pre, Post, Test, Inh, K_p, K_b, G, Pri, l >$, where:

- P is the finite set of places partitioned into a set of discrete places P_D, and a set of continuous places P_C, $P=P_D \cup P_C$, $P_D \cap P_C = \varnothing$. The discrete places may contain a natural number of tokens, while the marking of a continuous place is a real number (fluid level). In the graphical representation, a discrete place is drawn as a single circle while a continuous place is drawn with two concentric circles;
- T is a finite set of transitions, that can be partitioned into a set T_D of discrete transitions and a set T_C of continuous transitions, $T = T_D \cup T_C$, $T_D \cap T_C = \varnothing$. A discrete transition $t_j \in T_D$ is drawn as a black bar; a continuous transition $t_i \in T_C$ is drawn as a rectangle;
- *Pre*, *Test* and $Inh: P \times T \to Bag(P)$ respectively, are forward flow, test and inhibition functions. $Bag(P)$ is a discrete or real-valued multisets functions over P. The backward flow function in the multisets of P is $Post: T \times P \to Bag(P)$. These functions define the set of arcs A and it describes the marking-dependent cardinality of arcs connecting transitions with places and vice-versa. Also, the A set is partitioned into subsets: A_d, A_s, A_h, A_c and A_t. The subset A_d and A_s contains respectively the *discrete normal* and *continuous normal* arcs which can be seen as a function:

$$A_d: ((P_D \times T_D) \cup (T_D \times P_D)) \times IN_+^{|P|} \to IN_+,$$
$$A_s: ((P_C \times T_D) \cup \{T_D \times P_C)) \times IR^{|P|} \to IR.$$

The arcs of A_d and A_s, are drawn as single arrows. The subset of *discrete inhibitory* arcs is A_h: $(P_D \times T) \times IN_+^{|P|} \to IN_+$ or that of *continuous inhibitory* arcs A_h: $(P_C \times T) \times IR^{|P|} \to IR$. These arcs are drawn with a small circle at the end. The subset A_c defines the *continuous flow* arcs A_c: $((P_C \times T_C) \cup (T_C \times P_C)) \times IR^{|P|} \to IR$, and these arcs are drawn as double arrows to suggest a pipe.

A *test* input arc A_t is directed from a place of any kind to a transition of any kind, that A_t:$(P_D \times T) \times IN_+^{|P|} \to IN_+$ or A_t: $(P_C \times T) \times IR^{|P|} \to IR$ and is drawn as dotted single arrows. It does not consume the content of the source place. The arc of a net is drawn if the cardinality is not identically zero and it is labeled next to the arc, with a default value being 1. The IN_+ and IR are the sets of discrete natural and real numbers, respectively.

- $K_p: P_D \to IN_+$ is the capacity-function of discrete places and for each $p_i \in P_D$ this is represented by the minimum $K_{p_i}^{\min}$ and maximum $K_{p_i}^{\max}$ capacity which can contain an natural number of *tokens,* $0 \le K_{p_i}^{\min} \le K_{p_i}^{\min} < +\infty$;
- $K_b: P_C \to IR$ is the capacity-function of continuous places and for each $p_i \in P_C$ it describes the fluid lower bounds $x_i^{\min}$ and upper bounds $x_i^{\max}$ of the fluid, so that $-\infty < x_i^{\min} < x_i^{\max} < +\infty$. By default, the $x_i^{\min}$=0, $x_i^{\max}$ is $+\infty$;
- $G: T \times Bag(P) \to \{true, false\}$ is the *guard function* defined for each transition. For $t \in T$ a guard function $g(t, M)$ will be evaluated in each marking M, and if it evaluates to *true*, the transition may be enabled, otherwise t is disabled (by default is *true*);
- *Pri*: $T_D \to IN_+$ defines the priority functions for the firing of each transition. By default it is 0. The enabling of a transition with higher priority disables all the lower priority transitions;
- $l: T \cup P \to L$, is a labeling function that assigns a label to nodes (transitions and places) of a net in such a way that it maps the node name of the net into an action name or a condition name. ■

The structure of a *GDPN* is static. The dynamics of a net structure is specified by defining its initial marking and its marking evolution rule.

Definition 2: A timed marked labeled *GDPN* is a pair $NH = < N, M_0 >$, where $N = < H\Gamma, \theta, W, V >$ is a labeled *GDPN* structure (see Definition 1) with the respective attributes of timed transitions and M_0 is the initial marking of the net so that:

- The current marking (state) value of a net depends on the kind of place, and it is described by a pair of vector-columns $M = (\boldsymbol{m}, \boldsymbol{x})$, where $\boldsymbol{m}$: $P_D \rightarrow IN_+$ and $\boldsymbol{x}$: $P_C \rightarrow IR$ are the marking functions of respective type of places. $\boldsymbol{m} = (m_i p_i, m_i \geq 0, \forall p_i \in P_D)$ with $m_i p_i$ describe the number $m_i = \boldsymbol{m}(p_i)$ of tokens in discrete place p_i, and it is represented by black dots. $\boldsymbol{x} = (x_k b_k, x_k \geq x_k^{\min}, \forall b_k \in P_C)$ with $x_k b_k$ describe the fluid level $x_k = \boldsymbol{x}(b_k)$ in continuous place b_k and it is a real number, also allowed to take *negative* real value. The initial marking of net is $M_0 = (\boldsymbol{m}_0, \boldsymbol{x}_0)$. Vector-columns $\boldsymbol{m}_0$ and $\boldsymbol{x}_0$ give the initial marking of discrete places and of continuous places, respectively;
- The set of discrete transitions T_D is partitioned into $T_D = T_0 \cup T_\tau$, $T_0 \cap T_\tau = \varnothing$ so that: T_τ is a set of timed discrete transitions and T_0 is a set of immediate discrete transitions, so that $Pri(T_0) > Pri(T_\tau)$. A timed discrete transition $t \in T_\tau$ has a firing delay associated to it: $\theta : T_\tau \times Bag(P) \rightarrow IR_+$, which can be marking dependent. The IR_+ is the set of nonnegative real numbers. Timed discrete t is drawn as a black rectangle.
Let $T(M)$ denote the set of enabled transitions in current marking $M=(\boldsymbol{m}, \boldsymbol{x})$. Thus, a timed transition $t \in T_\tau(M)$ is enabled in current tangible marking M and it fires after a delay $\theta(t, M)$;
- $W : T_0 \times Bag(P) \rightarrow IR_+$ is the weight function of immediate discrete transitions $t_j \in T_0$, and this type of transitions is drawn with a black thin bar and has a zero constant firing time. Several enabled immediate transitions are scheduled to fire at the same time in *vanishing* marking M;
- $V : T_C \times Bag(P) \rightarrow IR$ is the marking dependent fluid rate function of timed continuous transitions T_c. If $t_i \in T_c$ is enabled in *tangible* marking M it fires with rate $V_i(M)$, so that it continuously changes the fluid level of place $b_i \in P_C$. ■

Figure 1 summarizes the all possible ways of placing arcs in a *GDPN* net for discrete transition and continuous transition with the discrete places and continuous places, respectively.

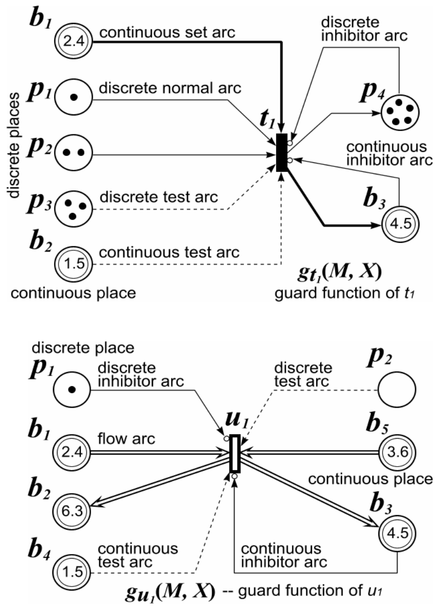

Fig. 1. All the possible ways of placing arcs in a timed *GDPN* net.

Enabling and firing of transitions is described in (Guţuleac, et al., 2006).

4. DESCRIPTIVE GDPN

In this section we present the model of *descriptive NH* net using the approach considered in (Guţuleac and Mocanu, 2005), which consists in incorporating the compositional features into *GDPN*. Due to the space restrictions we will only give a brief overview to this topic and refer the reader to (Guţuleac and Mocanu, 2005) and the references therein.

In following for abuse of notation, labels and name of transitions/places are the same.
A basic *descriptive expression* (*DE*) element (*bDE*) for a basic net (*bNH*) is:

$$bDE = |_{e_j}^{\alpha_j} \, m_i^0 p_i \, [W_i^+, W_i^-] \, |_{e_k}^{\alpha_k} .$$

The translation of this *bNH* is shown in figure 2a where e_j is the input event (transition t_j or rewriting rule r_j) with the action α_j and e_j is the output event e_k with the action α_k of the place p_i with initial marking $m_i^0 = M(p_i)$. The flow relation functions $W_i^+ = \mathrm{Pr}\,e(t_j, p_i)$ and $W_i^- = Post(t_k, p_i)$, return the multiplicities of the input and output arcs of p_i. The derivative elements of *bDE* are for $p_i^\bullet = \varnothing,\ W_i^- = 0$ is $|_{t_j}^{\alpha_j}\ m_i^0 p_i\ [W_i]$ with final place p_i of t_j and for $^\bullet p_i = \varnothing,\ W_i^+ = 0$ is $m_i^0 p_i\ W_i\ |_{t_k}^{\alpha_k}$ with entry place p_i of t_k (see figure 2b).

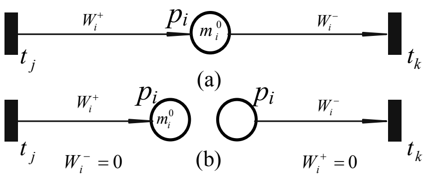

Fig. 2. Translation in *bPN* (a) of *bDE* and its derivatives (b).

If the initial marking m_i^0 of place p_i is zero tokens (or fluid level) we can omit $m_i^0 = 0$ in *bDE*. By default, if the type of action α is not mentioned this means it matches the name of a transition *t*. From a *bDE* we can build more complex *DE* of *NH* components by using composition operations.

The *DE* of *NH* is either a *bDE* or a composition of *DE*, i.e. *DE*::= *bDE* | $DE * DE$ | $\circ\, DE$, where $*$ represents any binary and $\circ$ any unary operation.

Some of them are the following. The unary *inhibition* "-" ($\overline{p}_i$) or *test* " ~"($\widetilde{p}_i$) operation describes the *inhibitory* or *test* arc, respectively. The $DE1 = m_i^0 \overline{p}_i\ [W_i]\,|_{t_j}^{\alpha_j}$ describe the inhibitor arc in *NH* with a weight function: $W_i = Inh(p_i, t_j)$.

The binary *sequential* operation "|" determines the logics of an interaction between two local states: pre-condition and post-condition, by the action (event). The specified conditions are always fulfilled.
The sequential operation is an *associative*, *reflexive* and *transitive* property, but is *not commutative.* The descriptive expression

$$DE\,2 = m_i^0 p_i\ [W_i]\,|_{t_j}^{\alpha_j}\, m_k^0 p_k\ [W_k]$$
$$\neq m_k^0 p_k\ [W_k]\,|_{t_j}^{\alpha_j}\, m_i^0 p_i\ [W_i]$$

means that the specified conditions (local state) associated with place-symbol p_i are always fulfilled and happen before the occurrence of the conditions associated with place-symbol p_k by means of the action t_j.

The binary *synchronization* operation is represented by the "$\bullet$" or "$\wedge$" operator which describes the rendez-vous synchronization (by the transition t_j) of two or more conditions represented respectively by symbol-place $p_i \in {}^\bullet t_j$, $i = \overline{1, n}$, i.e. it indicates that all preceding conditions of occurrence actions must have been completed.

The binary *split* operation represented by the "$\lozenge$" *split* operator describes and determines the causal relations between activity t_j and its post-conditions: after completion of the preceding action of t_j concomitantly several other post-condition can occur in parallel.

Thus the *NH* subnet shown in figure 1a and figure 1b is described by the *DE*'s *A* and *B*, respectively:
$A = A1\,|_{t_1}\ (p_4 \lozenge b_3)$; $B = B1\,|_{u_1}\ (6.35 b_2 \lozenge b_3)$ with
$A1 = (2.4 b_1 \cdot 1 p_1 \cdot 2 p_2 \cdot 3 \widetilde{p}_3 \cdot 1.5 \widetilde{b}_2 \cdot 5 \overline{p}_4 \cdot 4.5 \overline{b}_3)\,|_{t_1}$
and $B1 = (2.4 b_2 \cdot 1 \overline{p}_1 \cdot \widetilde{p}_2 \cdot 3.6 b_5 \cdot \widetilde{b}_4 \cdot 4.5 \overline{b}_3)\,|_{u_1}$

The compositional binary *competing parallelism* operation "$\vee$" means that it can be applied over two *NH* nets: NH_A with $DE_A = A$, and NH_B with $DE_B = B$. The resulting net NH_R with $DE_R = R$ can be represented by DE_R = R = A $\vee$ B. The fused nodes, with the same names, will inherit the incidence arcs of the marking-controlled nodes *A* and *B*.

The *precedence relations* between the operations in the *DE* is following: a) the evaluation of operations in *DE* are applied left-to-right; b) an unary operation binds stronger than a binary one; c) the "$\bullet$"operation is superior to"/ " and "$\lozenge$", in turn, these are superior to "$\vee$" operation. Further details on enabling and firing rules, and evolution for discrete part of *NH* can be found in (Guțuleac and Mocanu, 2005) as they require a great deal of space.

5. DYNAMIC REWRITING GDPN

In this section we introduce the model of *descriptive dynamic net rewriting systems.*
Let $X \rho Y$ be a binary relation. The *domain* of ρ is the $Dom(\rho) = \rho Y$ and the *codomain* of ρ is the $Cod(\rho) = X\rho$. Also, let $A=< Pre, Post, Test, Inh >$ be a set of arcs belonging to net $H\Gamma = < P, T, Pre, Post, Test, Inh, K_p, K_b, G, Pri, l >$ (see *Definition* 1).

Definition 3**:** A *dynamic rewriting GDPN* is a system $RH =< N, R, \phi, G_{tr}\ G_r, M >$, where:

- $N = < H\Gamma, \theta, W, V >$ and $R = \{r_1, ..., r_k\}$ is a finite set of discrete rewriting rules (DR) about the run-time structural modification of a net, so that $P \cap T \cap R = \varnothing$. In the graphical representation, the DR rule is drawn as two embedded empty rectangles;
- $\phi : E \to \{T_D, R\}$ is a function which indicates for every rewriting rule the type of event that can occur and $E = T_D \cup R$ denote the set of *events* of the net;
- $G_{tr} : R \times Bag(P) \to \{true, false\}$ is the *transition rule guard function* associated with $r \in R$, and $G_r : R \times Bag(P) \to \{true, false\}$ is the *rewriting rule guard function* defined for each rule of $r \in R$, respectively. For $\forall r \in R$, the function $g_{tr}(M) \in G_{tr}$ and $g_r(M) \in G_r$ will be evaluated in each marking and if they are evaluated to *true*, the rewriting rule r may be *enabled*, otherwise it is *disabled*. The default value of $g_{tr}(M) \in G_{tr}$ is *true* and for $g_r(M) \in G_r$ is *false,* in current marking M.

Let $R\Gamma =< N, R, \phi, G_{tr}, G_r >$ and the $RN =< R\Gamma, M >$ be represented by the descriptive expression $DE_{R\Gamma}$ and DE_{RN}, respectively. A dynamic rewriting structure modifying rule $r \in R$ of RN is a map $r : DE_L \rhd DE_W$, where the *codomain* of the $\rhd$ *rewriting operator* is a fixed descriptive expression DE_L of a subnet RN_L of current net RN, where the *domain* of the $\rhd$ is a descriptive expression DE_W of a new RN_W subnet. The rewriting operator $\rhd$ represents the binary operation which produces a *structure change* in the DE_{RN} and the net RN by replacing (rewriting) the fixed current DE_L of the subnet RN_L (DE_L and RN_L are dissolved) with the new DE_W of the subnet RN_W, now belonging to the new modified resulting $DE_{RN'}$ of the net $RN' = (RN \setminus RN_L) \cup RN_W$ where the meaning of $\setminus$ (and $\cup$) is operation of removing (adding) RN_L from (RN_W to) the net RN. In this new net RN', obtained by firing of enabled rewriting rule $r \in R$, the places and events with the same attributes which belong to RN' are fused.

A state configuration of a net RN is a pair $(R\Gamma, s)$, where $R\Gamma$ is the current structure of net together with a current state $s = (M, \beta(M))$. The $(R\Gamma_0, s_0)$ is called the initial state configuration of a net RN. ■

Enabling and Firing of Events. The enabling of events depends on the marking of all places. We say that a transition $t_j \in T_D$ of the event e_j is enabled in current marking M if the enabling condition $ec_d(e_j, M)$ and is verified. This is described in (Guţuleac, et al., 2006)

The discrete rewriting rule $r_j \in R$, that changes the structure of RN, is enabled in current marking M if the $ec_d(e_j, M)$ and the $g_{tr}(r_j, M)$ are verified.

Let $T_D(M)$ and $R(M)$, $T_D(M) \cap R(M) = \varnothing$, be the sets of enabled discrete transitions and enabled rewriting rule in current marking M, respectively. We denote the set of enabled events in a current marking M by the $E(M) = T_D(M) \cup R(M)$.

The event $e_j \in E(M)$ fires if no other event $e_k \in E(M)$ with higher priority is enabled. Hence, for each event e_j ***if*** $((\phi_j = t_j) \vee (\phi_j = r_j) \wedge (g_{tr}(r_j, M) = False))$ ***then*** the firing of transition $t_j \in T_D(M)$ or rewriting rule $r_j \in R(M)$ changes only the current marking:
$(R\Gamma, s) \xrightarrow{e_j} (R\Gamma, s') \Leftrightarrow (R\Gamma = R\Gamma$ and in $R\Gamma$ the $M[e_j > M'$). Also, for event e_j ***if*** $((\phi_j = r_j) \wedge (g_r(r_j, M) = True))$ ***then*** the event e_j occurs to firing of rewriting rule r_j and it changes the configuration and marking of the current net in the following way:

$$(R\Gamma, s) \xrightarrow{r_j} (R\Gamma', s'),\ M[r_j > M'.$$

The accessible state graph of a $RN = < R\Gamma, M >$ net is the labeled directed graph whose nodes are the states and whose arcs which are labeled with events or rewriting rules of RN :

a) *firing* of an enabled event $e_j \in E(M)$ determines an arc from the state $(R\Gamma, s)$ to the state $(R\Gamma, s')$ which is labeled with event e_j when this event can fire in the net configuration $R\Gamma$ at marking M and leads to a new state:

$$s' : (R\Gamma, s) \xrightarrow{e_j} (R\Gamma', s') \Leftrightarrow$$
$$(R\Gamma = R\Gamma' \text{ and } M[e_j > M' \text{ in } R\Gamma);$$

b) *change configuration*: arcs from state $(R\Gamma, s)$ to state $(R\Gamma', s')$ labelled with the rewriting rule $r_j \in R$, so that $r_j : (R\Gamma_L, M_L) \rhd (R\Gamma_W, M_W)$ which represent the change configuration of current RN net: $(R\Gamma, s) \xrightarrow{r_j} (R\Gamma', s')$ with $M[r_j > M'$.

As an example, let we consider the discrete part $RN1$ net given by the following descriptive expression:

$$DE_{R\Gamma 1} = p_1 |_{r_1} p_2 \vee \tilde{p}_1 |_{u_1} b_1[2.75] \vee DE'_{R\Gamma 1} \vee DE''_{R\Gamma 1},$$
$$DE'_{R\Gamma 1} = (p_2 \cdot p_5 \cdot b_1)|_{t_1} p_3 |_{t_2} p_4 |_{t3} (p_1 \Diamond p_5),$$
$$DE''_{R\Gamma 1} = (p_5 \cdot b_1[1.8])|_{u_2} b_2 \vee (\tilde{p}_4 \cdot b_1)|_{u_3},$$
$$M_0 = (5p_1, 1p_5, 12.5x_1), r_1 : DE_{R\Gamma 1} \rhd DE_{R\Gamma 2},$$
$$\beta(M_0) = diag(0.95, 0),\ g_r(r_1, M) = (m_1 = 3)\,\&\,(m_5 = 0).$$

The translation of the $DE_{R\Gamma 1}$ in $RN1$ is shown in figure 4a, and the $DE_{R\Gamma 2}$ in $RN\,2$ is shown in figure 4b.

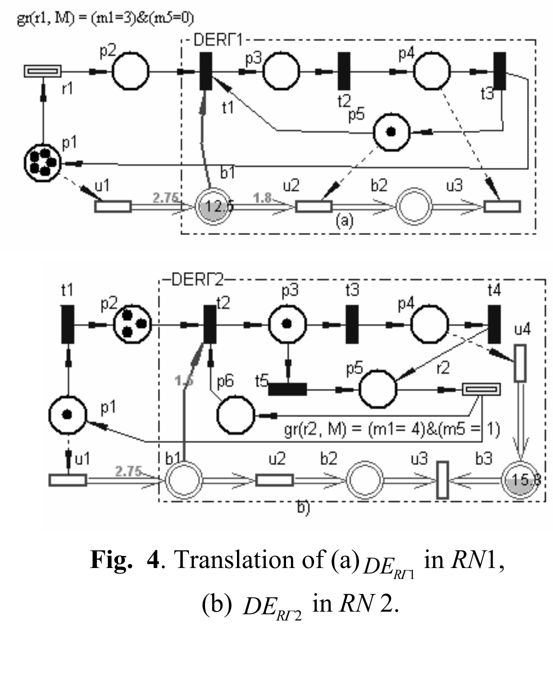

Fig. 4. Translation of (a) $DE_{R\Gamma 1}$ in $RN1$, (b) $DE_{R\Gamma 2}$ in $RN\,2$.

6. TIMED MEMBRANE GDPN

Also, for rewriting rule r_j is required to identify if RN_L net belongs to $R\Gamma$. Upon firing, the enabled events or rewriting rule modify the current marking.

Here we present the *DMH-nets* for encoding of hybrid P-systems mentioned above into descriptive dynamic rewriting *RN*. The basis for *DMH-nets* is a membrane structured *RN*, that the DE of *RN* net structure comprises: places and its capacity; transitions, and its priority and guard function; weighed directed arcs from places to transitions and vice-versa; weighed inhibitory and test arcs.

Consider the hybrid P-system ΓP. In order to represent the compartmentisation of membranes in ΓP with the RN we now use the notion of located places and located transitions introduced in (Guţuleac, 2005) and locally maximally concurrent executions of co-located transitions. The mapping of ΓP into *DMH-net* is constituted of two separate steps.

First, for every membrane $[_h\ ]_h$ we associate:

(i) to each object $\omega_{h,i} \in \omega_h$ one place $[_h\ y^0_{h,i} p_{h,i}]_h$ with the initial marking $y^0_{h,i} \in \{ m^0_h(p_i), x^0_h(b_i) \}$ of place $p_{h,i} \in P^h$ and to each continuous object rule $\rho^c_{h,j}$ one continuous transition $[_h\ u_{h,j}]_h$, $u_{h,j} \in T^h_C$ co-located in membrane h;

(ii) to each discrete object rule $\rho^{do}_{h,j}$ or discrete rewriting rules $\rho^{dm}_{h,j}$ one discrete event $[_h\ e_{h,j}]_h$, $e_{h,j} \in E^h$ that acts on this membrane.

Second, for every membrane $[_h\ ]_h$ we define the DE^0_h of RN^0_h that corresponds to the initial configuration of the P system ΓP as $[_h\ DE^0_h\ ]_h$.

Definition 4: The *DMH-nets* of degree $n \geq 0$, is a construct $DMH = \vee^{n-1}_{h=0} [_h\ DE_h\ ]_h$, where:

- The *evolving* object rule $\rho_{h',j} : [_h [_{h'}\ u \to v\ ]_{h'}]_h$ with multisets of objects u, v which will be kept in $[_{h'}\ ]_{h'}$ is encoded as: $[_h [_{h'}\ p_{h,u} |_{t_{h,j}} p_{h,v}\ ]_{h'}]_h$;
- The *antiport* rule with multisets of objects u, v and u', v', that realize a synchronization with object c and the exchange of the objects

$\rho_{h',i}:[_h u [_{h'} v]_{h'}]_h \rightarrow [_h v' [_{h'} u']_{h'}]_h$, is encoded as:

$$[_h [_{h'} (p_{h',u} \cdot p_{h',v} \cdot \tilde{p}_{h',c}) |_{t_{h',i}} (p_{h',u'} \Diamond p_{h',v'})]_{h'}]_h;$$

- The *symport* rule, that moves objects from inside to outside a membrane, or vice-versa $\rho_{h',k}:[_h u [_{h'}]_{h'}]_h \rightarrow [_h [_{h'} u']_{h'}]_h$ is encoded as: $[_h [_{h'} (p_{h',u} \cdot \tilde{p}_{h',c}) |_{t_{h',k}} p_{h',u'}]_{h'}]_h$;

Because the configuration means both a membrane structure and the associated multisets of objects, we need the discrete rewriting rules for processing of membranes' evolution and multisets of objects as:

$$MR = \{mr_0, mr_1, mr_2, mr_3, mr_4, mr_5, mr_6\}.$$

The above membrane rewriting rules (realized by the discrete rewriting events in *DE's*) are defined as:

- mr_0: *Change* rewriting rule, that in runtime the current structure and the multisets of objects to membrane *h*, encoded by descriptive expression $DE_{h'}$ and it marking $M_{h'}$ is changed in a new structure $DE'_{h'}$ with new marking $M'_{h'}$:
$[_h [_{h'} DE_{h'}]_{h'}]_h \rhd [_h [_{h'} DE'_{h'})]_{h'}]_h$;
- mr_1: *Dissolve* rewriting rule, that the objects as $M_{h'}$ and sub-membranes of membrane h' now belong to its parent membrane *h*:
$[_h DE_h [_{h'} DE_{h'}]_{h'}]_h \rhd [_h DE'_h]_h$,
$M'_h = M_h + M_{h'}$,
the skin membrane cannot be *dissolved*;
- mr_2: *Create* rewriting rule, that the new membrane h' with $DE''_{h'}$ and $M''_{h'}$, is created into membrane *h*, the rest remain in the parent membrane:
$$h:[_h DE_h]_h \rhd [_h DE'_h [_{h'} DE''_{h'}]_{h'}]_h,$$
$$M_h = M'_h + M''_{h'};$$
- mr_3: *Divide* rewriting rule, that the objects and sub-membranes are reproduced and added into membrane h' and h'', respectively:
$[_h DE_h]_h \rhd [_h [_{h'} DE_h]_{h'} [_{h''} DE_h]_{h''}]_h$;
- mr_4: *Merge* rewriting rule that the objects of membrane h' and h'' are added to a new membrane *h* is:
$[_h [_{h'} DE'_{h'}]_{h'} [_{h''} DE''_{h''}]_{h''}]_h \rhd [_h DE'_{h'} \vee DE''_{h''}]_h$
with the new marking $M'_{h'} + M''_{h''} = M_h$;
- mr_5: *Separate* rewriting rule is the counterpart of the *Merge* rewriting rule and is done by a form:
$[_h DE'_{h'} \vee DE''_{h''}]_h \rhd [_h [_{h'} DE'_{h'}]_{h'} [_{h''} DE''_{h''}]_{h''}]_h$
with meaning that the content of membrane h is split into two membranes, with labels h' and h'', and the new marking is $M_h = M'_h + M''_{h'}$;
- mr_6: *Move* rewriting rule where a membrane h'' can be moved out or moved into a membrane h' as a whole is done by a:
$[_h [_{h'} DE_{h'} [_{h''} DE_{h''}]_{h''}]_{h'}]_h [_h \rhd [_{h'} DE_{h'}]_{h'} [_{h''} DE_{h''}]_{h''}]_h$
with their markings, respectively. ■

Thus, using the *DMH-nets* facilitates a compact and flexible specification, verification and performance evaluation of parallel and distributed computing models of hybrid systems. In order to describe the details of this approach, we present a simple but illustrative example of encoding ΓP into *DMH-net*.

Consider the P system $\Gamma P1$ of degree 3 with the dissolving rule δ :

$$\mu = [_0\ b, [_1\ a, [_2\ ]_2,]_1,]_0,\ O_d = \{a,b,c,d\},$$
$$O_c = \{a_1^c, a_2^c, a_3^c\},\ \omega_0 = \{b, 15.8a^c 2\},\ \omega_2 = \varnothing,$$
$$\omega_1 = \{a, 10.5a_1^c\},\quad \rho_0^d = \{\rho_{0,1}^d : c \rightarrow dd_{in2},$$
$$\rho_{0,2}^d : b \rightarrow a_{2\,here}^c b_{in1}\},\ \rho_0^c = \{\rho_{1,1}^c : a_3^c \rightarrow a_{3\,out}^c\},$$
$$\rho_1^d = \{\rho_{1,1}^d : a \rightarrow b_{here} c_{out} d_{in2}, \rho_{1,2}^d : b \rightarrow a_{out}\delta,$$
$$\rho_{1,2}^c : b \rightarrow b_{here}\, 2.5a_{1\,here}^c\},\quad \rho_2^d = \{\rho_{1,1}^d : d \rightarrow b_{out1}\},$$
$$\pi_0 = \{\pi_{0,1} > \pi_{0,2}\},\quad \pi_1 = \varnothing,\quad \pi_2 = \varnothing.$$

The encoding solution for the initial configuration of $\Gamma P1$ is given by the *DM1-net*, where every object can be represented as a place labeled as the name of objects:

$$l(p_{0,1}) = l(p_{1,1}) = a,\ l(p_{0,2}) = l(p_{1,2}) = b,$$
$$l(p_{0,3}) = c,\ l(p_{2,1}) = l(p_{0,4}) = d,$$

and the number of tokens in this place denotes the number of occurrences of this object. Every object rule can be represented by an event type transition. For example, in membrane 0, the rule $\rho_{0,1} : c \rightarrow dd_{in2}$, can be described by a transition $t_{0,1}$. Because two copies of object *d* are send to membrane 2, the weight of the arc $(t_{0,1}, p_{2,1})$ is 2, which denotes that whenever the rule $\rho_{0,1}$ is performed, one copy of object *c* will be removed in membrane 0 and two copies of object *d* will be sent to membrane 2.

Up to now, all objects and rules of $\Gamma P1$ are encoded in *DMH1-net* as following:

$$DMH1 = [_0\ DE_0\ [_1\ DE_1\ [_2\ DE_2]_2\ ]_1\ ]_0,$$
$$DE_0 = p_{0,1} \vee p_{0,2} |_{t_{0,2}} (p_{2,1} \Diamond b_{0,1} |_{u_{0,1}}) \vee DE'_0,$$
$$DE'_0 = p_{0,3} |_{t_{0,1}} p_{2,1}[2],$$

$DE_1 = p_{1,1} \,|_{t_{1,1}}\, (p_{0,3} \Diamond p_{2,1} \Diamond p_{1,2} \,|_{r_{1,1}}\, p_{0,1} \vee DE_1')$,

$DE_1' = p_{1,2} \,|_{u_{1,1}}\, b_{1,1}[2.5] \,|_{u_{1,2}}\, b_{0,1})$,

$DE_2 = p_{2,1} \,|_{t_{2,1}}\, p_{1,2}$, $\Pr i(t_{0,1}) > \Pr i(t_{0,2})$,

$M = (1p_{0,2},\ 1p_{1,1},\ 15.8b_{0,1},\ 10.5b_{1,1})$,

$DE1 = DE_0 \vee DE_1 \vee DE_2$.

The dissolving rule $mr_1 = \delta$ is represented in *DMH1-net* by a following *Dissolve* rule:

$r_{1,1} : DMH\ 1 \triangleright DMH'1\ , DMH'1 = [_0\ DE_0'\ ]_0$,

$DE_0' = p_{0,1} \vee p_{0,3} \,|_{t_{0,1}}\, p_{0,4}[2] \vee DE1_0'$,

$DE1_0' = (p_{0,4} \cdot b_{0,1}) \,|_{t_{0,2}}\, (p_{0,2} \Diamond b_{0,2}) \vee DE2_0'$,

$DE2_0' = (\tilde{p}_{0,2} \cdot b_{0,1}) \,|_{u_{0,1}}\, b_{0,2} \,|_{u_{0,2}}\, b_{0,3} \,|_{u_{0,3}}\, b_{0,1}$,

$M = (1p_{0,1},\ 2p_{0,2}\ ,\ 2p_{0,4}, 25.6b_{0,1}, 10.5b_{0,3})$,

$gr(r_{1,1}, M^{DE1}) = (M^{DE1} = (1p_{0,3}, 2p_{1,2}, 1p_{2,1}))$,

where M^{DE1} is the current marking of $DMH\ 1$. The translation of $\Gamma P1$ into $DMH\ 1$ in figure 4a and $DMH'1$ *-net* is shown in figure 4b.

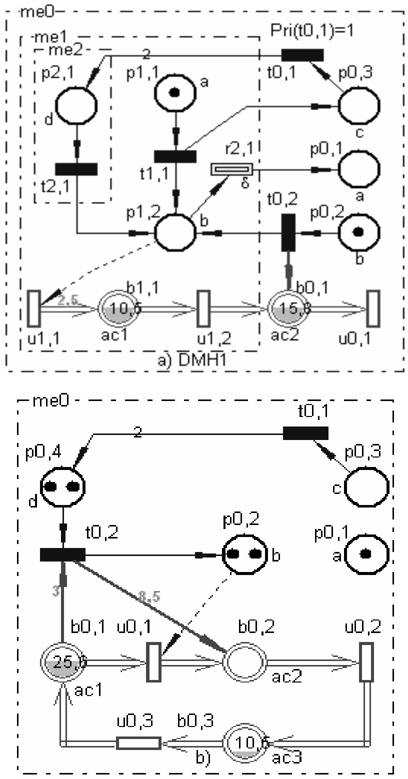

Fig. 4: Translation of $DE1$ into *DMH1-net* (a) and $DE'1$ into $DMH'1$ *-net* for $\Gamma P1$ (b).

The reachability graph of *DM1-net* in listing form is:

$M_0^{DE1} = (1p_{0,2}, 1p_{1,1})\, [U_1 > M_1^{DE1}$;

$M_1^{DE1} = (1p_{0,3}, 2p_{1,2}, 1p_{2,1})\, [U_2 > M_2^{DE'1}$;

$M_2^{DE'1} = (1p_{0,1},\ 2p_{0,2}, 2p_{0,4})\, [U_3 > M_3^{DE'1}$;

$M_3^{DE'1} = (1p_{0,1}, 3p_{0,2}, 1p_{0,4})\, [U_4 > M_4^{DE'1}$;

$M_4^{DE'1} = (1p_{0,1}, 4p_{0,2})\, [$, $U_1 = \{t_{0,2}, t_{1,1}\}$,

$U_2 = \{t_{0,1}, r_{1,1}, t_{2,1}\}$; $U_3 = U_4 = \{t_{0,3}\}$.

7. CONCLUSIONS AND FUTURE WORK

In this paper we have proposed a new approach for modeling performances of a P-system based on dynamic rewriting GDPN. In our method, components of continuous variant of P-systems are expressed through components of dynamic rewriting GDPN using descriptive expressions. For the creation of reconfigurable *GDPN* models we have also introduced a set of descriptive composition operations. In order to capture the localities and the behavior of reconfigurable and mobile hybrid systems, we defined the hybrid P-systems with active membranes and its formalization based on a new approach, the *DMH-nets* which allow for performance modeling.

Further work will be oriented towards the exploration of alternative modeling techniques based on Petri nets that have been suggested by their successful implementation in other scientific domains. Some features capturing the computational completeness of P-systems with maximal parallelism, and, in addition to the approaches proposed, stochastic techniques will be employed through Petri nets. Behavioral properties in P-systems such as *terminating*, *liveness*, and *boundedness* will be further included in our studies, based on these formalizations. We intend to gather our developments in the form of a high-level framework that includes proper modeling of the dynamic features of P systems (such as *dissolve*, *divide*, or *move*). The envisaged results may not be limited to P systems. We will also try to validate our modeling tool for both generic P-systems and dynamic GDPN, as a graphical as well as an algebraic tool.

We are currently developing a software visual simulator with a friendly interface for the verification and performance evaluation of descriptive rewriting *GDPN* and *DMH-nets*.

REFERENCES

[1] Alla, A., David, H., Continuous and Hybrid Petri Nets. Journal of Systems Circuits and Computers, 8 (1), pp. 159-188, 1998.

[2] Demongodin, I., and Koussoulas, N.T., Differential Petri Nets: Representing continuous systems in a discrete-event world, IEEE Transactions on Automatic Control, Vol. 43, No. 4, pp. 573-579, 1998.

[3] Guțuleac, E., Mocanu, M., and Țurcanu, I. Dynamic Rewriting of Differential Petri Nets for Modeling of Hybrid Systems, In Proceedings of the 2nd International Conference on Intelligent Computer Communication and Processing, 1-2 September 2006, Cluj-Napoca, România, Poster Volume, pp. 105-112.

[4] Guțuleac, E., and Mocanu, M. Descriptive Dynamic Rewriting GSPN-based Performance Modeling of Computer Systems, In: Proc. of the 15th International Conference CSCS15, 25-27 May 2005, Bucureşti, România, pp. 656-661.

[5] C. Martin-Vide and Păun, Gh., String Objects in P Systems. In Proceedings of Algebraic Systems, Formal Languages and Computations Workshop, 161-169, RIMS Kokyuroku, Kyoto University, 2000.

[6] Păun, Gh., Marcus Contextual Grammars. Kluwer, Dordrecht, 1997.

[7] Păun, Gh., Membrane Computing, An Introduction, Natural computing Series. ed. G. Rozenberg, Th. Back, A.E. Eiben, J.N. Kok, H.P. Spaink, Leiden Center for Natural Computing, Springer–Verlag, Berlin, pp. 420-428, 2002.

[8] Qi, Z., You, J., and Mao, H., P-Systems and Petri Nets, In Proceedings WMC 2003, Lecture Notes in Computer Science, vol. 2933, Springer-Verlag, Berlin, pp. 387-40