International Journal of Computers, Communications & Control
Vol. I (2006), No. 3, pp. 33-39

Descriptive Timed Membrane Petri Nets for Modelling of Parallel Computing

Emilian Guţuleac

Abstract: In order to capture the compartmentation and behaviour of membrane systems for modelling of parallel computing, we introduce the descriptive dynamic rewriting Descriptive Membrane Timed Petri Nets (DM-nets) that can at in run-time modify their own structure by rewriting some of their descriptive expression components. Furthermore, this descriptive approach facilitates the understanding of complex models and their component-based construction as well as the application of modern computer engineering concepts.
Keywords: Descriptive Petri nets, membrane systems, modelling, parallel computing.

1 Introduction

Recent technological achievements require advances beyond the existing computational models in order to be used effectively. Pragmatic aspects of current and future computer systems will be modelled so that realistic estimates of efficiency can be given for algorithms in these new settings.

Petri nets (PN) are very popular formalism for the analysis and representation of parallel and distributed computing in concurrent systems that has draw much attention to modelling and verification of this type of systems [1].

P systems, also referred to as membrane systems, are a class of parallel and distributed computing models [6]. The interest of relating P systems with the PN model of computation lead to several important results on simulation and decidability issues. Some efforts have been made to simulate P systems with Petri nets [2, 5, 7] to verifying the many useful behavioral properties such as reachability, boundedness, liveness, terminating, etc.

In this paper we propose a new approach to express the components of continuous-time P systems [6] throughout components of escriptive Petri Nets (PN) using descriptive expressions (DE) [3] for modelling of parallel computing. The DE are used for analytical representation and compositional construction of PN models. To model specific rules of P-systems within the framework of the descriptive Rewriting Timed PN (RTN) [4] we introduce a new extensions Ű the descriptive Membrane RTN, called DM-nets, that can modify dynamically their own structures by rewriting rules some of their components.

2 Labeled Extended Petri Nets

In this section, we define a variant of PN called labeled extended PN. Let L be a set of labels $L = L_P \uplus L_T$. Each place p_i labeled $l(p_i) \in P$ a local state and transition t_j has action labeled as $l(t_j) \in L_T$.

A labeled extended PN is structure as a $\Gamma =< P, T, Pre, Post, Test, Inh, G, Pri, K_p, l >$, where: P is the finite set of places and T is a finite set of transitions that $P \cap T = \emptyset$. In the graphical representation, the place is drawn as a circle and the transition is drawn as a black bar; The Pre, $Test$ and $Inh : P \times T \times \mathcal{N}^{|\mathcal{P}|} \to \mathcal{N}_+$ respectively is a forward flow, test and inhibition functions and is a backward flow function in the multi-sets of P, where defined the set of arcs A and describes the marking-dependent cardinality of arcs connecting transitions and places. The set A is partitioned into tree subsets: A_d, A_h, and A_t. The subset A_d contains the directed arcs which can be seen as A_d: $((P \times T) \cup (T \times P)) \times \mathcal{N}^{|\mathcal{P}|} \to \mathcal{N}_+$ and are drawn as single arrows. The inhibitory arcs $A_h : (P \times T) \times \mathcal{N}^{|\mathcal{P}|} \to \mathcal{N}_+$are drawn with a small circle at the end. The test arcs $A_t : (P \times T) \times \mathcal{N}^{|\mathcal{P}|} \to \mathcal{N}_+$are directed from a place to a transition, and are drawn as

Selected paper from ICCCC 2006

dotted single arrows. It does not consume the content of the source place. The arc of net is drawn if the cardinality is not identically zero and this is labeled next to the arc and by a default value being 1; $G : E \times \mathcal{N}^{|\mathcal{P}|} \to \{true, false\}$ is the guard function transitions. For $t \in T$ a guard function $g(t,M)$ that will be evaluated in each marking, and if it evaluates to *true*, the transition t may be enabled, otherwise t is disabled (the default value is *true*); $Pri : T \to \mathcal{N}_+$ defines the priority functions for the firing of each transition that maps transitions onto natural numbers representing their priority level. The enabling of a transition with higher priority disables all the lower priority transitions; $K_p : P \to \mathcal{N}_+$ is the capacity of places, and by default being infinite value; The $l : T \cup P \to L$, is a labeling function that assigns a label to a transition and places. In this way that maps transition name into action names that $l(t_j) = l(t_k) = \alpha$ but $t_j \neq t_k$ and $l(p_i) = l(p_n) = \beta$ but $p_i \neq p_n$.

A marked labeled extended PN net is a pair $N =< \Gamma, M_0 >$, where Γ is a labeled PN structure and M_0 is the initial marking of the net. $M : P \to \mathcal{N}_+$ is the current marking of net which is described by a symbolic vector-column $M = (m_i p_i), m_i \geq 0, \forall p_i \in P$, where the $(m_i p_i)$ is the number m_i of tokens in place p_i. The M is the state of net that assigns to each place tokens, represented by black dots.

The details concerning on enabling and firing rules, and evolution for of $N =< \Gamma, M_0 >$ can be found in [3] as they require a great deal of space.

3 Descriptive expressions of Petri nets

Due to the space restrictions we will only give a brief overview to this topic and refer the reader to [3, 4] and the references therein. In following for abuse of notation, labels and name of transitions/places are the same. We use the concept of a basic descriptive element (*bDE*) for a basic PN (*bPN*) introduced in [2] as following: $bDE = |_{t_j}^{\alpha_j} m_i^0 p_i [W_i^+, W_i^-] |_{t_k}^{\alpha_k}$. The translation of this *bPN* is shown in figure 1a, where respectively is input transition (action type α_j) and $t_k = p_i^\bullet$ is the output transition (action type α_k) of place $p_i \in P$ with initial marking m_i^0, and the flow type relation functions $W_i^+ = Pre(t_j, p_i)$ and $W_i^- = Post(t_j, p_i)$, respectively which return the multiplicity of input and output arcs of the place $p_i \in P$. The derivative elements of *bDE* are for $p_i^\bullet = \emptyset, W_i^- = 0$ is $|_{t_j}^{\alpha_j} m_i^0 [W_i]$ with final place p_i of t_j and ${}^\bullet p_i = \emptyset, W_i^+ = 0$ is $m_i^0 p_i [W_i] |_{t_k}^{\alpha_k}$ with entry place p_i of t_k. If the initial marking m_i^0 of place is a zero tokens we can omit m_i^0 in *bDE*. By default, if the type of action α is not mentioned this to match the name of a transition t. From a *bDE* we can build more complex DE of PN components by using composition operations. Also by default,if $W_i^+ = W_i^- = 1$, we present *bDE* and it derivatives as following:$|_{t_j}^{\alpha_j} m_i^0 p_i |_{t_k}^{\alpha_k}$, $|_{t_j}^{\alpha_j} m^0 p_i$ or $m_i^0 p_i |_{t_k}^{\alpha_k}$.

A descriptive expression (*DE*) of a labeled PN is either *bDE* or a composition of *DE* a *N*: $DE ::= bDE | DE * DE | \circ DE$, where $*$ represents any binary composition operation and $\circ$ any unary operation.

Descriptive Compositional Operations. In the following by default the labels of *N* are encoded in the name of the transitions and places. The composition operations are reflected at the level of the *DE* components of *N* models by fusion of places, fusion of transitions with same type and same name (label) or sharing of as subnets.

Place-Sequential Operation. This binary operation, denoted by the " | " *sequential operator*, determines the logic of a interaction between two local states p_i (pre-condition) and p_k (post-condition) by t_j action that are in precedence and succeeding (causality-consequence) relation relative of this action. Sequential operator is the *basic mechanism* to build *DE* of *N* models. This operation is an *associative*, *reflexive* and *transitive* property, but is *not commutative* operation. The means the fact $DE1 = m_i^0 p_i [W_i] |_{t_j}^{\alpha_j} m_k^0 p_k [W_k] \neq m_k^0 p_k [W_k] |_{t_j}^{\alpha_j} m_i^0 p_i [W_i]$ that the specified conditions (local state) associated with place-symbol p_i are fulfilled always happens before then the occurrence of the conditions associated with place-symbol p_k by means of the action t_j. Also, the PN modelling of the *iteration* operation is obtained by the fusion of head (entry) place with the tail (final) place that are the same name (*closing* operation) in *DE* which describes this net. The self-loop of $N2$ net described by an:

$DE2 = m_i^0 p_i[W_i]|_{t_j}^{\alpha_j} p_i[W_i] = m_i^0 \tilde{p}_i[W_i]|_{t_j}^{\alpha_j}$, it is the test operator "$\tilde{p}$", i.e. represent the *test* arc. The translation of $DE2$ in $N2$ is shows in figure 2b.

Inhibition Operation. This unary operation is represented by inhibitory operator "- " (place-symbol with overbar) and it $DE3 = m_i^0 \bar{p}_i[W_i]|_{t_j}^{\alpha_j}$ describe the inhibitor arc with a weight $W_i = Inh(p_i, t_j)$.

Synchronization Operation. This binary operation is represented by the " $\bullet$" or " $\wedge$"*join* operator describe the rendez-vous synchronization (by transition t_{t_j}) of a two or more conditions represented respectively by symbol-place $p_i \in^\bullet t_j, i = \overline{1,n}$, i.e. it indicate that all preceding conditions of occurrence actions must have been completed. This operation is a commutative, associative and reflexive.

Split Operation . This binary operation represented by the " $\diamondsuit$" *split* operator and it describe 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 take occurs in parallel ("message sending"). Property of split operation is a commutative, associative and reflexive.

Competing Parallelism Operation. This compositional binary operation is represented by the " $\vee$" competing parallelism operator, and it can be applied over two N_A with $DE_A = A$ and N_B with $DE_B = B$ or internally into resulting N_R with $DE_R = R$, between the places of a single N_R which the symbol-places with the same name are fused, respectively. We can represent the resulting $DE_R = A \vee B$ as a set of ordered pairs of places with the same name to be fused, with the first element belonging to A the second to B. The fused places will inherit the arcs of the place in A and B . Also, this compositional binary operation is a *commutative*, *associative* and *reflexive* property.

Precedence Relations between the Operations. We introduce the following precedence relation between the compositional operations in the DE: 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 " $\diamondsuit$", in turn, its are superior the "$\vee$ " operation. Further details on definitions, enabling and firing rules, and evolution for of N can be found in [3] as they require a great deal of space.

4 Dynamic Rewriting Petri Nets

In this section we introduce the model of *descriptive dynamic net rewriting* PN system. Let $X\rho Y$ is a binary relation. The *domain* of is the $Dom(\rho) = \rho Y$and the *codomain* of ρ is the $Cod(\rho) = X\rho$. Let $A =< Pre, Post, Test, Inh >$ is a set of arcs belong to netΓ .

A descriptive dynamic rewriting PN system is a structure $RN =< \Gamma, R, \phi, G_{tr}, G_r, M >$,
where: $=< P, T, Pre, Post, Test, Inh, G, Pri, K_p, l >$; $R = r_1, ..., r_k$ is a finite set of rewriting rules about the runtime structural modification of net that $P \cap T \cap R = \emptyset$. In the graphical representation, the rewriting rule is drawn as a two embedded empty rectangle. We let $E = T \cup R$ denote the set of events of the net; $\phi : E \to T, R$ is a function indicate for every rewriting rule the type of event can occur; $G_{tr} : R \times \mathcal{N}^{|\mathcal{P}|} \to \{true, false\}$ and $G_r : R \times \mathcal{N}^{|\mathcal{P}|} \to \{true, false\}$ is the *transition rule guard function* associated with $r \in R$ and the rewriting rule guard function defined for each rule of $r \in R$, respectively. For $\forall r \in R$, the $g_{tr} \in G_{tr}$ and $g_r \in G_r$will be evaluated in each marking and if its are evaluates to *true*, the rewriting rule r may be *enabled*, otherwise it is disabled. Default value of $g_{tr} \in G_{tr}$ is *true* and for $g_r \in G_r$ is *false*. Let $RN =< R\Gamma, M >$ and $R\Gamma =< \Gamma, R, \phi, G_{tr}, G_r >$ described with 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 whose *codomain* of the rewriting operator $\rhd$ is a fixed descriptive expression DE_L of a subnet RN_L of current net RN, where $RN_L \subseteq RN$,with $P_L \subseteq P$, $E_L \subseteq E$and set of arcs $A_L \subseteq A$ and whose *domain* of the $\rhd$ is a descriptive expression DE_W of a new RN_W subnet with $P_W \subseteq P$, $E_W \subseteq E$ and set of arcs A_W. The $\rhd$ rewriting operator represent binary operation which produce a *structure change* in the DE_{RN} and the net RN by replacing (rewriting) of the fixed current DE_L of subnet RN_L (DE_L and RN_L are dissolved) by the new DE_W of subnet RN_W now belong to the new modified resulting $DE_{RN'}$ of net $RN^{'} = (RN \backslash RN_L) \cup RN_W$ with $P^{'} = (P \backslash P_L) \cup P_W$ and $E^{'} = (E \backslash E_L) \cup E_W$, where $A^{'} = (P \backslash P_L) \cup A_W$ the meaning of $\backslash$ (and $\cup$) is

operation to removing (adding) RN_L from ($RN_W$to) net RN. In this new net $RN^{'}$, obtained by execution (fires) of enabled rewriting rule $r \in R$, the places and events with the same attributes which belong $RN^{'}$ are fused, respectively. By default the rewriting rules $r : DE_L \rhd \emptyset$ and$r : \emptyset \rhd DE_W$ describe the rewriting rule which fooling holds $RN^{'} = (RN \backslash RN_L)$ and $RN^{'} = (RN \cup RN_W)$, respectively. A state of a net RN is a pair $(R\Gamma, M)$, where $R\Gamma$ is the configuration of net together with a current marking M. Also, the pair $(R\Gamma_0, M_0)$ with $P_0 \subseteq P$, $E_0 \subseteq E$ and marking M_0 is called the initial state of the net.

Enabling and Firing of Events. The enabling of events depends on the marking of all places. We say that a transition t_j of event e_j is enabled in current marking M if the following enabling condition $ec(t_j, M)$ is verified:

$ec(t_j, M) = (\wedge_{\forall p_i \in {}^\bullet t_j}(m_i \geq Pre(p_i, t_j)) \wedge (\wedge_{\forall p_k \in {}^\circ t_j}(m_k < Inh(p_i, t_j))) \wedge (\wedge_{\forall p_l \in {}^* t_j}(m_l \geq Test(p_l, t_j))) \wedge (\wedge_{\forall p_n \in t_j^\bullet}((K_{p_n} - m_i) \geq Post(p_n, t_j)))) \wedge g(t_j, M))$.

Similarly, the rewriting rule $r_j \in R$ is enabled in current marking M if the following enabling condition $ec_{tr}(r_j, M)$is verified:

$ec_{tr}(r_j, M) = (\wedge_{\forall p_i \in {}^\bullet r_j}(m_i \geq Pre(p_i, r_j)) \wedge (\wedge_{\forall p_k \in {}^\circ r_j}(m_k < Inh(p_i, r_j))) \wedge (\wedge_{\forall p_l \in {}^* r_j}(m_l \geq Test(p_l, r_j))) \wedge (\wedge_{\forall p_n \in r_j^\bullet}((K_{p_n} - m_i) \geq Post(p_n, r_j)))) \wedge g(r_j, M))$.

Let the $T(M)$ and $R(M)$ is respectively the set of enabled transitions and rewriting rule in current marking M. Let the $E(M) = T(M) \uplus R(M)$, is the set of enabled events in a current marking M. The event $e_j \in E(M)$ fire if no other event $e_k \in E(M)$ with higher priority has enabled. Hence, for e_j event $if((\phi_j = t_j) \vee (\phi_j = r_j) \wedge (g_{tr}(r_j, M) = false))$ *then* (the firing of transition $t_j \in T(M)$ or rewriting rule $r_j \in R(M)$ change only the current marking:$(R\Gamma, M) \stackrel{e_j}{\rightarrow} (R\Gamma, M^{'}) \Leftrightarrow (R\Gamma = R\Gamma^{'}$ and $M[e_j > M^{'}))$. Also, for e_j event $if((\phi_j = r_j) \wedge (g_r(r_j, M) = true)$ *then* (the event e_j occur to firing of rewriting rule r_j and it occurrence change configuration and marking of current net:$(R\Gamma, M) \stackrel{r_j}{\rightarrow} (R\Gamma^{'}, M^{'})$, $M[r_j > M^{'}$).

The accessible state graph of a net$RN =< \Gamma, M >$ is the labeled directed graph whose nodes are the states and whose arcs which is labeled with events of RN are of two kinds: a) firing of a enabled event $e_j \in E(M)$: arcs from state $(R\Gamma, M)$ to state$(R\Gamma, M^{'})$ labeled with evente_j then this event can fire in the net configuration $R\Gamma$ at marking M and leads to new marking $M^{'}$: $(R\Gamma, M) \stackrel{e_j}{\rightarrow} (R\Gamma^{'}, M^{'}) \Leftrightarrow (R\Gamma = R\Gamma^{'}$ and $M[e_j > M^{'}$ in $R\Gamma)$; b) change configuration: arcs from state $(R\Gamma, M)$ to state($R\Gamma^{'}, M^{'}$) labeled with rewriting rule r_j :$(R\Gamma_L, M_L) \rhd (R\Gamma_W, M_W)$ which represent the change configuration of current RN net: $(R\Gamma, M) \stackrel{e_j}{\rightarrow} (R\Gamma^{'}, M^{'})$ and $M[r_j > M^{'}$.

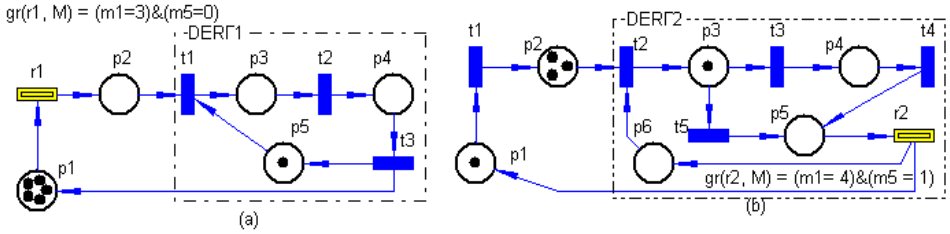

Figure 1: Translation of (a) $DE_{R\Gamma 1}$in $RN1$ and (b) $DE_{R\Gamma 2}$ in $RN2$

Let we consider the $RN1$ given by the following descriptive expression:$DE_{R\Gamma 1} = p_1|_{r_1} p_2 \vee DE^{'}_{R\Gamma 1}$, $DE^{'}_{R\Gamma 1} = (p_2 \cdot p_5)|_{t_1} p_3|_{t_2} p_4|_{t_3}(p_1 \diamondsuit p_5)$, $M_0 = (5p_1, 1p_5)$, $g_r(r_1, M) = (m_1 = 3)\&(m_5 = 0)$ and $r_1 : DE_{R\Gamma 1} \rhd DE_{R\Gamma 2}$. Also, for r_j is required to identify if RN_L belong the $R\Gamma$. Upon firing, the enabled events or rewriting rule modify the current marking and/or and modify the structure and current marking of net $RN1$ in $RN2$ given by: $DE_{R\Gamma 2} = p_1|_{t_1} p_2 \vee DE^{'}_{R\Gamma 2}$, $DE^{'}_{R\Gamma 2} = (p_2 \cdot p_6)|_{t_2} p_3(|_{t_3} p_4|_{t_4} p_5 \vee |_{t_5} p_5|_{r_2}(p_1 \diamondsuit p_6))$, $M = (1p_1, 3p_2, 1p_3)$, $g_r(r_2, M) = (m_1 = 4)\&(m_5 = 1)$, $r_2 = r_1^{-1} : DE_{R\Gamma 2} \rhd DE_{R\Gamma 1}$.

Figure 1 show the translation of $DE_{R\Gamma 1}$ in $RN1$ and $DE_{R\Gamma 2}$ in $RN2$, respectively.

5 Dynamic Rewriting Timed Petri Nets

Systems are described in timed PN (TPN) as interactions of components that can performed a set of activities associated with events. An event $e = (\alpha, \theta)$, where $\alpha \in E$ is the type of the activity (action name), and θ is the firing delay.

A descriptive dynamic rewriting TPN as a $RTN =< RN, \theta >$, where: $RN =< \Gamma, R, \phi, G_{tr}, G_r, M >$, $\Gamma =< P, T, Pre, Post, Test, Inh, G, Pri, Kp, l >$ (see Definition 2 and 3) with set of events E which can be partitioned into a set E_0 of *immediate* events and a set E_τ of *timed* events $E = E_0 \uplus E_\tau$. The immediate event is drawn as a thin bar and timed event is drawn as a black rectangle for transition or a two embedded empty rectangle for rewriting rules, and $Pri(E_0) > Pri(E_\tau)$; $\theta : E \times \mathbb{N}^{|\mathcal{P}|} \to \mathcal{R}_+$ is the weight function that maps events onto real numbers $\mathcal{R}_+$ (delays or weight speeds). Its can be marking dependent. The delays $\theta(e_k, M) = d_k(M)$defining the events firing parameters governing its duration for each timed events of E_τ. If several timed events are enabled concurrently $e_j \in E(M)$ for $e_j \in {}^\bullet p_i = \forall e_j \in E : Pre(p_i, e_j) > 0$, either in competition or independently, we assume that a race race competition condition exists between them. The evolution of the model will determine whether the other timed events have been aborted or simply interrupted by the resulting state change. The $\theta(e_j, M) = w_j(M)$ is weight speeds of immediate events $e_{j \in E_0}$. If several enabled immediate events are scheduled to fire at the same time in *vanishing* marking M with the weight speeds, and the probability to enabled immediate event e_j can *fire* is: $q_i(M) = w(e_j, M) / \sum_{e_l \in (E(M) \& {}^\bullet p_i)} w(e_j, M)$,where $E(M)$ is the set of enabled events in M. An immediate events $e_j \in T_0$ has a zero firing time.

6 P Systems and Descriptive Timed Membrane Petri Nets

Here we give a brief review of P systems and its encoding with DM-nets. The main components of P systems are membrane structures consisting of membranes hierarchically embedded in the outermost skin membrane. A full guide for P systems can be referred to [3]. In general, a basic evolution-communication P system with *active membranes* (of degree $n \geq 0$) is $\Pi = (O, H, \mu, \Omega, (\rho, \pi))$, where: O is the alphabets of objects; H is a finite set of labels for membranes;μ is a membrane structure consisting of n membranes labeled with elements h in H;Ω is the configuration, that is a mapping from membranes of Π(nodes in μ) to multisets of objects $\omega_k \in \Omega, k = 1, , |\Omega|$, from O;ρ and π is respectively the set off developmental rules ρ_h and $\pi_h$its priorities , $h = 0, 1, , n-1$. Thus the can be of two forms of rules: a) the *object rules* (OR), i.e., evolving and communication rules concerning the objects; b) the membranes rules (MR), i.e., the rules about the structural modification of membranes.

Here we define DM-Nets for encoding of P systems mentioned above into descriptive dynamic rewriting TPN as a RTN. The basis for DM-Nets is a membrane RTN that is DE net structure comprise: places; transitions; weighed directed arcs from places to transitions and vice-versa; a capacity for each place; weighed inhibitory and test arcs; priority and guard function of transitions.

The $DM - nets$ of degree $n \geq 0$ is a construct $DM = \vee_{h=0}^{n-1} [_h DE_h]_h$, where DE_h is the descriptive expression of RTN_h that represent the configuration of membrane $[_h \,]_h$ in a P system Π.

Consider the P system Π.The encoding of Π into RTN_Π is decomposed into two separate steps. First, for every membrane $[_h \,]_h$ we associate: to each object $\omega_i \in \Omega$ one place $p_{h,i} = [_h m_i^0 \, p_i]_h$ labeled as ω_i with the initial marking m_i^0, and to each rule $\rho_{h,j} \in \rho$ one event $e_{h,j} = [_h e_j]_h$ labeled as $\rho_{h,j}$ that acts on the this membrane. Second, for every membrane $[_h \,]_h$ we define the DE_h of RTN_h that it correspond to the initial configuration of the P system Π as $[_h DE_h]_h$.

Let u, v , and $u_\prime, v_\prime$, is a multiset of objects. The *evolving* object rule $\rho_{h',j} : [_h [_{h'} u \to v]_{h'}]_h$ with multiset of objects u, v , which will be kept in membrane $[_h]_h$ is encoded as $[_h [_{h'} \, p_u |_{t_j} p_v]_{h'}]_h$. The antiport rule $\rho_{h',j} : [_h u [_{h'} v]_{h'}]_h \to [_h v' [_{h'} u']_{h'}]_h$, that realize a synchronized wich object c the exchange of objects, is encoded as $[_h [_{h'} (p_u \cdot p_v \cdot \tilde{p}_c) |_{t_j} (p_{u'} \diamondsuit p_{v'})]_{h'}]_h$. Also, the symport rule $\rho_{h',k} : [_h u [_{h'}]_{h'}]_h \to [_h [_{h'} u']_{h'}]_h$ that

move objects from inside to outside a membrane, or vice-versa is encoded as $[_h[_{h'}(p_u \cdot \tilde{p}_c)|_{t_k} p_{u'}]_{h'}]_h$.

Because a configuration mean both a membrane structure and the associated multisets, we need rules for processing membranes and multisets of objects as:

$MR = Change, Dissolve, Create, Divide, Merge, Separate, Move.$

The above membrane rewriting rules (realized by the rewriting events in DE) are defined as follows:

*Change*rewriting rule $[_h[_{h'}(DE_{h'}, M_{h'})]_{h'}]_h \rhd [_h[_{h'}(DE^{'}_{h'}, M^{'}_{h'})]_{h'}]_h$ that in runtime the current structure and the multisets of objects to membrane h, encoded by descriptive expression $DE_{h,}$ and marking $M_{h,}$ is changed in a new structure $DE^{'}_{h,}$ with new marking $M^{'}_{h,}$;

Dissolve rewriting rule $[_h(DE_h, M_h)[_{h'}(DE_{h'}, M_{h'})]_{h'}]_h \rhd [_h(DE_h, M^{'}_h)]_h$ that the objects and sub-membranes of membrane $h^{'}$ now belong to its parent membrane h , the skin membrane cannot be dissolved;

Create rewriting rule $[_h(DE_h, M_h)]_h \rhd [_h(DE^{'}_h, M^{'}_h)[_{h'}(DE^{''}_{h'}, M^{''}_{h'})]_{h'}]_h$ with $M_h = M^{'}_h + M^{''}_{h'}$ that the new membrane $h^{'}$ is created and $M^{''}_{h'}$ are added into membrane $h^{'}$, the rest remain in the parent membrane h; Divide rewriting rule $[_h(DE_h, M_h)]_h \rhd [_h[_{h'}(DE_h, M_h)]_{h'}[_{h''}(DE_h, M_h)]_{h''}]_h$that the objects and sub-membranes are reproduced and added into membrane $h^{'}$ and membrane $h^{''}$, respectively;

Merge rewriting rule that the objects of membrane $h^{'}$ and $h^{''}$ are added to a new membrane h is: $[_h[_{h'}(DE^{'}_{h'}, M_{h'})]_{h'}[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_h \rhd [_h(DE^{'}_{h'} \vee DE^{''}_{h''}, M_{h'} + M^{''}_{h''})]_h$;

Separate rewriting rule is the counterpart of *Merge* is done by a rewriting rule of the form $\rhd [_h(DE^{'}_{h'} \vee DE^{''}_{h''}, M_{h'} + M^{''}_{h''})]_h \rhd [_h[_{h'}(DE^{'}_{h'}, M_{h'})]_{h'}[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_h$ with the meaning that the content of membrane h is split into two membranes, with labels $h^{'}$ and $h^{''}$.

*Move*rewriting rule where a membrane $h^{''}$ can be moved out or moved into a membrane $h^{'}$ as a whole is: $[_h[_{h'}(DE_{h'}, M_{h'})[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_{h'}]_h \rhd [_h[_{h'}(DE_{h'}, M_{h'})]_{h'}[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_h$ or

$[_h[_{h'}(DE_{h'}, M_{h'})]_{h'}[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_h \rhd [_h[_{h'}(DE_{h'}, M_{h'})[_{h''}(DE^{''}_{h''}, M^{''}_{h''})]_{h''}]_{h'}]_h$.

Thus, using the $DM-Nets$ facilitates a compact and flexible specification to visual simulate of P systems with dynamic rewriting TPN nets that permit the verification of the its many useful behavioral properties such as reachability, boundedness, liveness, terminating, etc., and the performance evaluation of parallel computing models.

7 Summary and Conclusions

In this paper we have proposed an approach to the performance modeling of the behaviour of P-systems through a class of Petri nets, called Descriptive Membrane Timed PN (DM-nets). Based upon the introduction of a set of descriptive composition operation and rewriting rules attached with transitions for the creation of dynamic rewriting TPN, the membrane structure can be successfully encoded as a membrane descriptive rewriting timed Petri nets models which permit the description the behavioral state based process run-time structure change of P systems. We are currently developing a software visual simulator with a friendly interface for verifying and performance evaluation of descriptive rewriting TPN models and DM-nets.

References

[1] M. Ajmone-Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Francheschinis, “Modeling with Generalized Stochastic Petri Nets,” *ser. In Parallel Computing* ,New York: Wiley, 1995.

[2] S. Dal Zilio, E. Formenti, “On the Dynamics of PB System: a Petri Net View,” *In Proceedings WMC 2003, Lecture Notes in Computer Science 2933* , Springer-Verlag, pp. 153-167, 2004.

[3] E. Gutuleac, "Descriptive Compositional Construction of GSPN Models for Performance Evaluation of Computer Systems," *In Proceedings of the 8-th International Symposium on Automatic Control and Computer Science, SACCS2004, 22-23 October*,Iasi, Romania, CD, 2004.

[4] E. Gutuleac, "Descriptive Dynamic Rewriting GSPN-based Performance Modeling of Computer Systems," *Proceedings of the 15th International Conference on Control Systems and Computer Science, CSCS15, 25-27 May 2005*,Bucuresti, Romania, pp. 656-661, 2005.

[5] J. Kleijn, M. Koutny, G. Rozenberg, "Towards a Petri Net Semantics for Membrane Systems," *In Proceedings of the WMC6 2005, July 18-21*, Wien, Austria, pp. 439-459, 2005.

[6] Gh. Paun, "Membrane Computing. An Introduction," *Natural computing Series. ed. G. Rozenberg, Th. Back, A.E. EibenJ.N. Kok, H.P. Spaink, Leiden Center for Natural Computing*, Springer-Verlag, Berlin, p. 420, 2002.

[7] Z. Qi, J. You, and H. Mao, "P Systems and Petri Nets," *Proceedings WMC 2003, Lecture Notes in Computer Science, vol. 2933*, Springer-Verlag, Berlin, pp. 387-403, 2003.

Emilian Guţuleac,
Technical University of Moldova,
Computer Science Department,
Address: 168, Bd. Stefan cel Mare, MD-2004,
Chişinău, Republic of Moldova
E-mail: egutuleac@mail.utm.md