

VHPN – SOFTWARE TOOL FOR VISUAL DISCRETE-CONTINUOUS MODELLING OF HYBRID SYSTEM USING GENERALIZED TIMED DIFFERENTIAL PETRI NETS

Emilian GUȚULEAC, Iurie ȚURCANU, Emilia GUȚULEAC, Diana ODOBESCU

*Computer Science Department, Technical University of Moldova
Bul. Stefan cel Mare nr. 168, MD-2012 Chisinau, Republic of Moldova
egutuleac@mail.utm.md*

Abstract. *This paper presents the timed Generalized Differential Petri Nets (GDPN) a class of Petri nets, that are accepted the negative-continuous place capacity, negative real values for continuous place marking and negative marked-dependent arc cardinalities. A Visual Hybrid Petri Nets (VHPN) software tool that offers a graphical user interface has been developed and described for visual simulation of GDPN. The continuous transitions of the net are fired with speeds that are piecewise constants or marked-dependent variable.*

Keywords: *Generalized Differential Petri Nets, hybrid systems, software tool, modelling.*

1. Introduction

Researches in the area of specification, modeling and simulation of parallel hybrid systems aim at verification, coherence and speed up of systems development, starting from requirement capture, high level and low level specification up to final prototyping. These goals are achieved through verification, translation and refinement of formal models. They can be applied to large dynamic information and computer systems.

Among the formalisms that are used, Petri nets (PN) are the most popular. A number of different classes of PN have been proposed [1, 2, 3, 4, 5]. Hybrid Petri nets (HPN) are PN based models in which some places may hold a discrete number of tokens and places a continuous quantity represented by a real number. Timed HPN were introduced in [3], were extended to include non-determinism both for firing times and rate of continuous transitions. Enabled continuous and discrete transitions may then fire according to their firing speeds or time delays, respectively.

To make design issues and procedures more transparent, we tried to deviate as little as possible from the concepts and tenets of HPN. Thus, we created our extension building on relevant previous works: HPN, and Generalized Timed-T Discrete Petri nets [2]. The result is a new type of HPN with we call *Generalized Differential Petri Net (GDPN)*, and is able to represent continuous systems and discrete systems in a common

model. The novel features of GDPN are accepted the negative-continuous place capacity, negative real values for continuous place marking and negative token-dependent arc cardinalities, that permit to generalize the concept of HPN and Differential Petri Nets [5].

The GDPN structure is formed by three kinds of discrete and continuous objects: places, transitions and arcs. Places represent some kind of resources, transitions denote actions or events that happen in the system (and arcs link the first two kinds of objects together, implementing the logic of the GDPN; they assign actions to resources, and vice versa. Certainly this is only a basic set of objects. Though their structures are simple, analysis of HPN is very complicated thing. There are several groups all over the world working on analysis of Petri nets, and certainly there is a bunch of modelling software packages [6,7,8]. Most tools provide a graphical user interface for the convenient drawing and editing of the model. They usually include analysis components for one or several classes of models. But still there is not such a program that would satisfy everyone's needs, both commercial and educational, so we decided to start developing our own software for modeling GDPN, and called it the Visual Hybrid Petri Nets (VHPN).

The paper is organized as follows. In Section 2 we describe the model definition of generalized timed GDPN. Subsequently in Section 3 we considered the VHPN tool overview. Visual

simulation facility of timed *GDPN* in *VHPN* is presented in Section 4. Conclusions are drawn in Section 5.

2. Generalized Timed Differential Petri nets

Various extensions have been made to the framework of timed *HPN* since its in [1, 2, 3, 5]. The definition of generalized *GDPN* with customary notation is derived from [2, 3, 5].

Definition 1. A *GDPN* structure is a 14-tuple $HF = \langle P, T, Pre, Post, Test, Inh, K_p, K_b, G, Pri, M_0, \theta, W, V \rangle$, 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 , $P = P_D \cup P_C$, $P_D \cap P_C = \emptyset$. 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, $T \cap P = \emptyset$, that can be partitioned into a set T_D of discrete transitions and a set T_C of continuous transition, $T = T_D \cup T_C$, $T_D \cap T_C = \emptyset$. A discrete transition $t_j \in T_D$ is drawn as a black bar and continuous transition $t_i \in T_C$ is drawn as a rectangle;

- $Pre, Test$ and $Inh: P \times T \rightarrow Bag(P)$ respectively is a forward flow, test and inhibition functions. $Bag(P)$ is a discrete or continuous multiset over P . The backward flow function in the multisets of P is a $Post: T \times P \rightarrow Bag(P)$, where define the set of arcs A and it describes the marking-dependent cardinality of arcs connecting transitions with places and vice-versa. The A is partitioned into subsets: A_d, A_s, A_h, A_c and A_t . The subset A_d contains the *discrete normal* arcs which can be seen as a function $A_d: ((P_D \times T_D) \cup (T_D \times P_D)) \times IN_+^{|P|} \rightarrow IZ$ and the $A_s: ((P_C \times T_D) \cup (T_D \times P_C)) \times IR^{|P|} \rightarrow IR$ the *continuous set* of arcs. The arcs of A_d and A_s , are drawn as single arrows. The subset of *discrete inhibitory* arcs is $A_h: (P_D \times T) \times IN_+^{|P|} \rightarrow IN_+$ or *continuous inhibitory* arcs which is $A_h: (P_C \times T) \times IR^{|P|} \rightarrow IR$. These arcs are drawn with a small circle at the end. The subset A_c

defines the *continuous flow* arcs $A_c: ((P_C \times T_C) \cup (T_C \times P_C)) \times IR^{|P|} \rightarrow IR$, 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 $A_t: (P_D \times T) \times IN_+^{|P|} \rightarrow IN_+$ or $(P_C \times T) \times IR^{|P|} \rightarrow IR$ and are drawn as 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 it is labeled next to the arc with a default value being 1.

The IZ, IN_+ and IR are the set of discrete integer, natural and real numbers, respectively;

- $K_p: P_D \rightarrow IN_+$ is the function-capacity of discrete places and for each $p_i \in P_D$ this is represented by the minimum $K_{p_i}^{\min}$ and maximum capacity $K_{p_i}^{\max}$, $0 \leq K_{p_i}^{\min} < +\infty < K_{p_i}^{\max}$ which can contain an *integer* number of *tokens*, respectively. By default $K_{p_i}^{\min} = 0$ and $K_{p_i}^{\max}$ being infinite value;

- The $K_b: P_C \rightarrow IR$ is the function-capacity of continuous places and for each $p_i \in P_C$ describes the fluid lower bound x_i^{\min} and upper bounds x_i^{\max} of fluid on each continuous place, that $-\infty < x_i^{\min} < x_i^{\max} < +\infty$. This x_i^{\max} by default it is ∞ , and bound has no effect when it is set to infinity. Each continuous place has an implicit lower bound at level is 0;

- $G: T \times Bag(P) \rightarrow \{True, False\}$ 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 (the default value is *true*);

- $Pri: T_D \rightarrow IN_+$ defines the priority functions for the firing of each transition. The enabling of a transition with higher priority disables all the lower priority transitions;

- The current marking (state) value of a net depends on the kind of place, and it is described by a pair of vectors $M = (\mathbf{m}, \mathbf{x})$, where the $\mathbf{m}: P_D \rightarrow IN_+$ and $\mathbf{x}: P_C \rightarrow IR$ are marking functions of respectively type of places. The vector-column

$\mathbf{m}=(m_i p_i, m_i \geq 0, \forall p_i \in P_D)$ with $m_i p_i$ is the number m_i of tokens in discrete place, and it is represented by the black dots. The $\mathbf{x}=(x_k b_k, x_k \geq x_k^{\min}, \forall b_k \in P_C)$ is vector-column, where $x_k b_k$ is the fluid level x_k in continuous place b_k , and it is the real number, that is allowed to take also negative real value.

The initial marking of net is $M_0=(\mathbf{m}_0, \mathbf{x}_0)$. The vector \mathbf{m}_0 gives the initial marking of discrete places and the vector \mathbf{x}_0 gives the initial marking of fluid places;

- The set of discrete transitions T_D is partitioned into $T_D = T_\tau \cup T_i$, $T_\tau \cap T_i = \emptyset$ so that: T_τ is a set of timed discrete transitions and T_i is a set of immediate discrete transitions. The $Pri(T_i) > Pri(T_\tau)$. Let $T(M)$ denote the set of enabled transitions in current marking $M=(\mathbf{m}, \mathbf{x})$.

A timed discrete transition $t \in T_\tau$ is drawn as a black rectangle and has a firing delay $\theta: T_\tau \times Bag(P) \rightarrow IR_+$ is associated to it, and this is can be marking dependent. Thus, a timed transitions $t \in T_\tau(M)$ is enabled in current tangible marking M , it fires after delay $\theta(t, M)$. Note once again, we do allow the firing delay to be dependent on fluid levels;

- $W: T_i \times Bag(P) \rightarrow IR_+$ is the weight function of immediate discrete transitions $t_j \in T_i$, and this type of transition is drawn with a black thin bar and has a zero constant firing time. If several enabled immediate transitions $t_j \in T_i(M)$ are scheduled to fire at the same time in *vanishing* marking M , the transitions t_k with the respective weights w_k fire with probability:

$$q(t_k, M) = w(t_k, M) / \sum_{t_j \in T_i(M)} w(t_j, M);$$

- $V: T_c \times Bag(P) \rightarrow IR_+$ is the marking dependent fluid rate function of timed continuous transitions T_c . These rates appear as labels next to the continuous timed transitions. If $t_i \in T_c$ is enabled in *tangible* marking M it fires with rate $V_i(M)$, that continuously change the fluid level of continuous place P_C . ■

We denote the set of all *markings* (or the partially discrete and partially continuous state space) of the net by $S = IN_+^{|P_D|} \times IR^{|P_C|}$. In the fol-

lowing we denote by S_D and S_C , $S = S_D \cup S_C$, $S_D \cap S_C = \emptyset$ are the discrete and the continuous component of the state space, respectively. The current marking $M=(\mathbf{m}, \mathbf{x})$ evolves in time. We denote the time by τ , and can think the marking $M(\tau)$ at time τ as the marking process $S(\tau) = \{ \mathbf{m}(\tau), \mathbf{x}(\tau), \tau > 0 \}$ of net.

Upon firing, the discrete (respectively continuous) transition removes a specified number (quantity) of tokens (fluid) for each discrete (fluid) input place, and deposits a specified number (quantity) of tokens (fluid) for each discrete (fluid) output place. The levels of fluid places can change the enabling/disabling of continuous transitions;

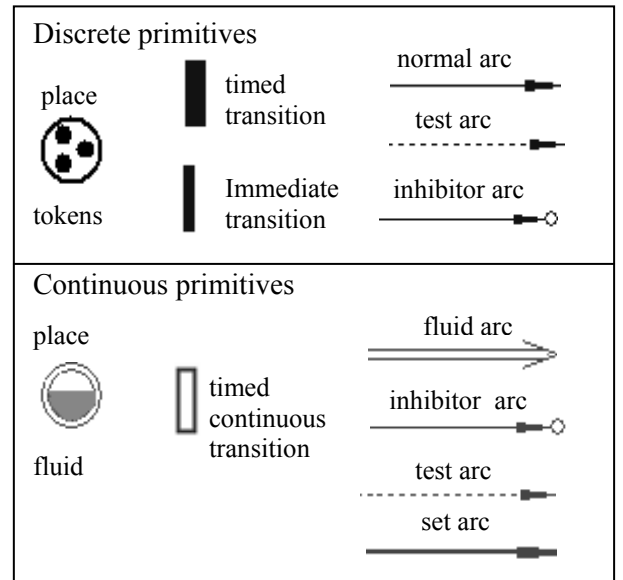


Figure 1. All the primitive of the GDPN formalism.

Figure 1 summarizes the graphical representation of all the *HPN* primitives.

The GDPN has five kinds of arcs; normal discrete (continuous set) input/output arcs, flow (continuous) input/output arcs, discrete (continuous) inhibitory input arcs and discrete (continuous) test input arcs. A normal discrete (flow or continuous set) input arc (flow) is directed to a discrete (continuous) transition from a discrete place (flow place) from which it consumes respectively a specified number of tokens (quantities of fluid) of the source place by firing of transition. An inhibitory and test input arc is di-

rected from a place of any kind to a transition of any kind. It does not consume the content of the source place. These five arcs are called input arcs. A normal discrete (flow) output arc is directed from a discrete (continuous) transition to a discrete (flow) place which it deposits a specified number of tokens (quantities of fluid) in each respectively output discrete (flow) place by firing of transition.

Figure 2a and Figure 2b summarize the all possible ways of placing arcs in a *GDPN* net for discrete transition and continuous transition.

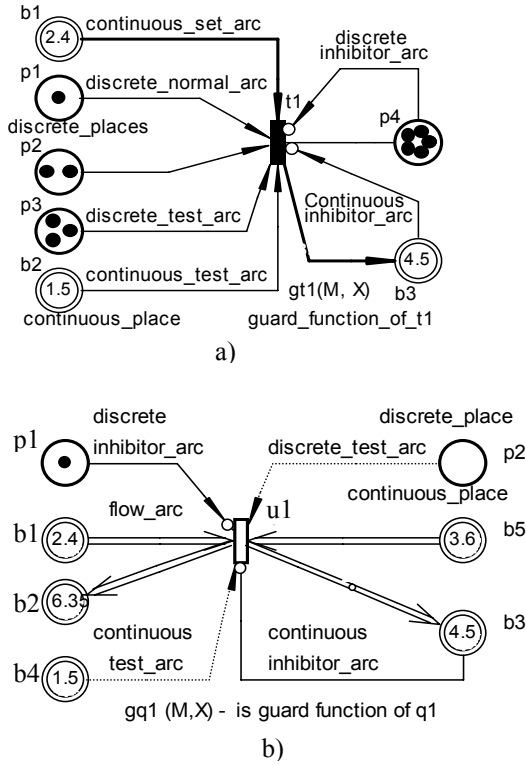


Figure 2. All the possible ways of placing arcs in a net.

The role of the previous set and functions will be clarified by providing the enabling and firing rules. Let us denote by m_i the i -th component of the vector \mathbf{m} , i.e., the number of tokens in discrete place p_i when the marking is \mathbf{m} , (and x_k denote the k -th component of the vector \mathbf{x} , i.e., the fluid level in fluid place p_k when the continuous marking is \mathbf{x}).

Enabling and Firing of Transitions. Given $t_j \in T$, we denote ${}^{\circ}t_j = \{p_i \in P : Pre(p_i, t_j) > 0\}$ the input and $t_j^* = \{p_i \in P : Post(p_i, t_j) > 0\}$ output set places of t_j , and with

${}^{\circ}t_j = \{p_i \in P : Inh(p_i, t_j) > 0\}$ the inhibition and ${}^*t_j = \{p_i \in P : Test(p_i, t_j) > 0\}$ the test set places of transition t_j , respectively. Also, in following we denote by $u_k \in T_C$ the continuous transitions and by $b_k \in P_C$ the continuous places can be distinct it between discrete transitions and continuous places, respectively.

Let $T(M)$ the set of enabled transitions in current marking M . We say that a discrete transition $t_j \in T_D(M)$ is enabled in current marking M if the following logic expression (enabling condition $ec_d(t_j)$) is verified:

$$ec_d(t_j) = \left(\bigwedge_{\forall p_i \in {}^{\circ}t_j} (m_i \geq Pre(p_i, t_j)) \right) \& \left(\bigwedge_{\forall p_k \in {}^*t_j} (m_k < Inh(p_k, t_j)) \right) \& \left(\bigwedge_{\forall p_l \in {}^*t_j} (m_l \geq Test(p_l, t_j)) \right) \& \left(\bigwedge_{\forall p_n \in t_j^*} ((K_p - m_n) \geq Post(p_n, t_j)) \right) \& \left(\bigwedge_{\forall b_i \in {}^{\circ}t_j} (x_i \geq Pre(b_i, t_j)) \right) \& \left(\bigwedge_{\forall b_k \in {}^*t_j} (x_k < Inh(b_k, t_j)) \right) \& \left(\bigwedge_{\forall b_l \in {}^*t_j} (x_l \geq Test(b_l, t_j)) \right) \& \left(\bigwedge_{\forall b_n \in t_j^*} ((K_b - x_n) \geq Post(x_n, t_j)) \right) \& g(t_j, M) .$$

The transition $t_j \in T_D(M)$ fire if no other transition $t_k \in T_D(M)$ with higher priority has enabled. Also, we say that a continuous transition $u_j \in T_C(M)$ is enabled and continuously fire in current marking M if the following logic expression (the enabling condition $ec_c(u_j)$) is verified:

$$ec_c(u_j) = \left(\bigwedge_{\forall b_i \in {}^{\circ}u_j} (x_i > 0) \right) \& \left(\bigwedge_{\forall p_k \in {}^*u_j} (m_k < Inh(p_k, u_j)) \right) \& \left(\bigwedge_{\forall p_l \in {}^*u_j} (m_l \geq Test(p_l, u_j)) \right) \& \left(\bigwedge_{\forall b_k \in {}^*u_j} (x_k < Inh(b_k, u_j)) \right) \& g(t_j, M) \& \left(\bigwedge_{\forall b_l \in u_j} (x_l \geq Test(b_l, u_j)) \right) \& \left(\bigwedge_{\forall b_n \in u_j^*} ((K_b - x_n) \geq V_j \cdot Post(x_n, u_j)) \right)$$

and no transition with higher priority has concession.

If an immediate discrete transition has concession in marking $M = (\mathbf{m}, \mathbf{x})$, it is enabled and the marking is vanishing. Otherwise, the marking is tangible and any timed discrete transition with concession is enabled in it.

An immediate discrete transition t_j enabled in marking $M=(\mathbf{m}, \mathbf{x})$ yields a new vanishing marking $M'=(\mathbf{m}', \mathbf{x})$. We can write $(\mathbf{m}, \mathbf{x}) [t_j > (\mathbf{m}', \mathbf{x})$. If the marking $M=(\mathbf{m}, \mathbf{x})$ is tangible, fluid could continuously flow through the flow arcs A_c of enabled continuous transitions into or out of fluid places. As a consequence of this, a transition t_c is *enabled* at M iff for every $p_c \in \bullet t_c$, $\mathbf{x}(p_c) > 0$, and its *enabling degree* is:

$$\text{enab}(t_c, M) = \min_{p_c \in \bullet t_c} \{\mathbf{x}(p_c) / \text{Pre}(p_c, t_c)\}.$$

In the *GDPN*, the potential rate $\beta_i(M)$ of change of fluid level in place $p_i \in P_C$ in marking current M is given by:

$$\beta_i(M) = \sum_{t_k \in T(M)} [V_{k,i}(M) - V_{i,k}(M)],$$

where for any given $t_k \in T_C$, $V_{k,i}(M)$ is an input fluid rate of fluid place $p_i \in P_C$ and $V_{i,k}(M)$ is an output fluid rate of this place. We allow the firing rates and the enabling functions of the timed discrete transitions, the firing speeds and enabling functions of the timed continuous transitions, and arc cardinalities to be dependent on the current state of the *GDPN*, as defined by the marking $M(\tau)$.

In the following, we describe the Visual Hybrid Petri Nets (*VHPN*) software tool for visual simulation of *GDPN* models.

3. VHPN tool overview

VHPN is a window-based, object-oriented tool, in which elements typical of hybrid Petri net models (places, transitions, and arcs) are manipulated under the assistance of basic syntactical rules that prevent the construction of incorrect models. This tool is designed with a broad functionality: it contains a large number of various extensions to stochastic and timed hybrid Petri nets that have been proposed, user-defined place and transition rewards, place capacities, discrete and continuous arc marking-dependent multiplicities, firing marking-dependent firing discrete and flows rates, single and infinite-server transitions, and memory policies. The model drawn using the interface is internally translated into several files that are however completely transparent to the user.

The representation of *GDPN* model is thus the object to which all the command provided by the Graphical User Interface (*GUI*). The *GUI* is shown in Figure 3.

Having specified the *GDPN* models, it can be saved into a data-base from which it can be retrieved for any subsequent use.

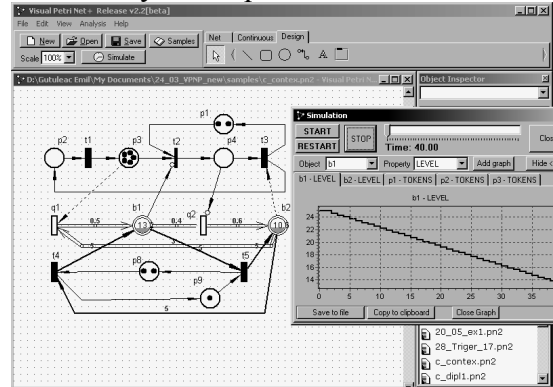


Figure 3. Graphical User Interface of VHPN.

The simulation of a *GDPN* model is performed in *VHPN* by first constructing the model on the screen of a graphical workstation using the facilities provided by the graphical user interface. Once that the simulation model has been specified, a debugging phase must take place to insure that it is first syntactically, and then semantically correct. Finally, the actual simulation is performed of the model are collected and analyzed to obtain estimates for the performance indexed that are relevant for the study.

Constructing *GDPN* models consists of drawing places and transitions connected by arcs. This is made possible by *VHPN* that provides a *drawing board* where graph nodes of *GDPN* selected from a menu of basic objects can be deposited. To facilitate the drawing of graphically well organized models, a grid can be superimposed on its corners. Discrete direct (fluid and continuous set) arcs that can be of input, output, and discrete (continuous) test and inhibition type connect nodes. They are drawn selecting first the origin node, and subsequently the destination node. Arcs behave as rubber-bands and can look like segmented lines though the use of intermediate points. Objects deposited on the drawing board can be repositioned by dragging them around with usual point-and-select methods. Arcs stick to the origin and destination nodes so that moving nodes affects connecting arcs as

well. To make the model look nicer, segmented arcs can be turned into continuous (curved) lines by means of a *spline* option that smooths their angles. Group of elements (subnets) can be graphically manipulated as a single object using a multiple selection option which then allows to move all the elements of a group. Selected groups of elements can be removed from the drawing with a single operation.

An important aspect of timed *GDPN* models is the specification of the initial marking of the net that is relevant both as the starting point for the future simulation experiment, and as the specification that provides additional meaning to certain elements of the net and thus identifies the set of possible states of the model. The initial marking is obtained editing a place element parameter to specify the number of tokens or fluid level that is initially contained in each type place. A default value of zero tokens is introduced to make the specification of the initial marking quicker. The specification of the model is completed with the definition of the parameters and of the performance indexes relevant for the study. All the transitions of the net possess default parameter values that can be modified with simple editing actions. When indications exist that the model will be used for several studies in which the effect of the variation of certain parameters will be investigated, these parameters can be specified as variables, using the *VHPN* object inspector.

4. Visual simulation of timed *GDPN*

The specification of a model using a *GDPN* based formalism has the advantage of an explicit representation of the relations among the different models of the net (interconnections) and of the activities that are currently taking place (tokens and/or fluid). Moreover the dynamics of the model is completely represented by the movement of such tokens. Advanced graphics allows exploit these advantages recognizing their importance in the description of a model and in the understanding of its behaviour.

Visual simulation is a method that exploits the graphical description of the model to show the dynamic evolution that takes place during its simulation directly on the graphical representa-

tion. Static graphical facilities are used to describe the topology of the *GDPN* model. The dynamic graphical facilities are used to visualize the movement of the tokens (fluid), to provide snapshots of the model, and to represent the evaluation of computed performance. Visual simulation is very useful during the early stage of the construction of a model since it represents a powerful tool for its debugging. Moreover, the animation of a correct simulation model may provide important insights on the behavior of the actual system, especially because of the possibility of highlighting particular evolution patterns that may be crucial for the good (or bad) behavior of the system, but difficult to deduce from average performance indexes obtained with standard methods. The simulation is a stochastic experiment in which is performed to achieve necessary data. Also in this mode the user can see consecutive change of markings in the nets, exalted transitions and other aspects of Petri net principles, which makes this analysis mode more educative and attractive for those new to Petri nets formalism.

VHPN at the moment is still in the development stage and some its modules, or even most of its structure can undergo considerable changes. But the idea of building a user-friendly, handy tool for analyzing stochastic and timed *GDPN*. Algorithms of functioning of the editor of a *GDPNs* and the inspector of objects are tool complex and insignificant in a theme of the given degree work and consequently here to not be considered. Let's consider algorithm of functioning of a simulator of a network. Feigning represents the process of a net in real time, also in a mode of accelerated or slowed down time. Feigning begins with initial marks (which it is restored after end of process). By means of process of simulation it is possible evidently representing the process of transition of a timed *GDPN* of one condition in another, also to check correctness of functioning and to collect the various information in real-time. As some transitions can be in the conflict before the operation, of each following transition there is a check on his opening.

The computation of the marking-dependent flow rate across a continuous arc (the continuous throughput), requires a more detailed analyses.

A problem arises when the fluid level reaches one of the boundary conditions. Consider the input fluid places $b_i \in \bullet u_j$ and the output fluid places $b_k \in u_j^\bullet$ of a continuous transition $u_j \in T_C$, and the actual fluid levels of this places are respectively $x_i^{min} < x_i < x_i^{max}$ and $x_k^{min} < x_k < x_k^{max}$. We require that for every transition $u_j \in T_C$:

$$\delta_j = \min_{\forall b_i} \{ (x_i - x_i^{min}) / Pre(b_i, u_j) \} \text{ and}$$

$$\varepsilon_j = \min_{\forall b_k} \{ (x_k - x_k^{max}) / Post(u_j, b_k) \},$$

$$\eta_j = \min_{\forall b_k} \{ \delta_j / \varepsilon_j \}. \text{ The actual fluid rate } V_j^* \text{ of}$$

a continuous transition $u_j \in T_C$ is determinate in the following: if $(\eta_j < V_j)$ then $V_j^* = \eta_j$ else $V_j^* = V_j$, where V_j is potential flow rate function of this continuous transitions.

In the case of structural and effective conflicts, $|b_i^\bullet| > 1$ these are solved using the *priority semantic* or *proportional semantic* [1].

For example we consider the fluid place $b_i \in P_C$ with fluid level $x_i^{min} < x_i < x_i^{max}$, multiple inputs $\{u_2, u_4\} \in \bullet b_i$, and outputs $\{u_1, u_3\} \in b_i^\bullet$ continuous transitions with the potential flow rate V_j .

Let $\alpha_j = V_j \cdot Post(u_j, b_i)$ and $\beta_k = V_k \cdot Pre(u_k, b_i)$, $i = 2, 4$ and $k = 1, 3$. If $\alpha_2 + \alpha_4 < \beta_1 + \beta_3$, the fluid level as function of the time decreases with rate $(\alpha_2 + \alpha_4) - (\beta_1 + \beta_3)$, until it reaches the boundary value $x_i = x_i^{min}$. From this point on, the potential flow rate is set to 0 and place b_i remains constantly empty. In order to compute the flow throughput along the arcs, the individual flow rates along each arc must be redefined in such a way that the difference between input and output rates remains equal to 0. By renaming the new output flow rates $\beta_1^* + \beta_3^*$, different semantics can be adopted [1] to redistribute the flow rate such the following relation is verified: $(\alpha_2 + \alpha_4) = (\beta_1^* + \beta_3^*)$. Under a priority semantic, the input flows are used to saturate the output flows in a predefined order. If for example the transition u_1 has priority over the u_3 , $Pri(u_1) > Pri(u_3)$, then: if $(\alpha_2 + \alpha_4) \leq \beta_1$,

then $\beta_1^* = \alpha_2 + \alpha_4$ and $\beta_3^* = 0$; if $\beta_1 < \alpha_2 + \alpha_4 \leq \beta_1 + \beta_3$ then $\beta_1^* = \beta_1$ and $\beta_3^* = \alpha_2 + \alpha_4 - \beta_1$.

Under a proportional semantic instead, the input flow is proportioned in away that keeps constant the ratio between the output flows, that is:

$$\beta_k^* = \beta_k \cdot (\alpha_2 + \alpha_4) / (\beta_1 + \beta_3), k = 1, 3.$$

Simulation is started from the separate form of a Figure 4.

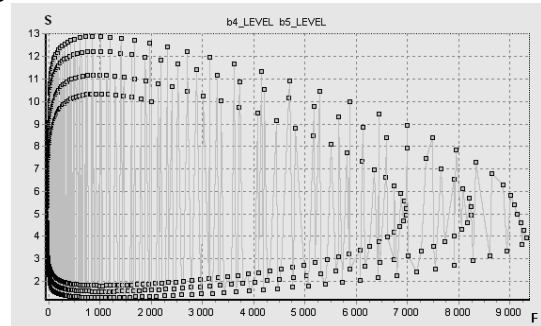


Figure 4. The form of simulation in VHPN.

The description of the form: START - start of a network on simulation; RESTART - start of a network on simulation with restoration of initial marking; STOP - stops of simulation, can be continued by button START; Time - modeling time; the Toddler - sets speed of simulation; Close - to close a Window. The panel for creation of diagrams as a chosen parameter is displayed in the diagram. For a choice of parameter all over again it is necessary to choose an object, then his parameter and press the button *Add graph*. For displaying a part of the form with diagrams press "Show >>>". If the diagrams are not necessary at present it is possible to hide them, pressing "Hide <<<" to a Figure 4. If the diagram is used for inserting into a report or in another program, it can be done with the help of the buttons "Save to file" - preservation of the diagram in a file of a format *name.bmp* or copy to clipboard - to copy in the buffer of an exchange. The Simulator supports all types of objects realized in the program.

The step of time simulation gets out automatically, on the following rule: The *minimal step* = *MAXINT*; For all transitions the beginning: If transition time that $temp = 1/Rate$; If transition continuous that $temps = 1$; If $temp < Minimal\ step$ then *Minimal step* = $temp$; The end.

However, it may occurrence the situations when only immediate transitions are enabled. Then time

costs on a place and if in this case there is even one enabled timed transition, it will not be fired. When a discrete or continuous transition fires, tokens or fluid will be symbolically moved along the input arcs from the input places to the transition and along the output arcs from the transition to the output places, according to the various arc multiplicities. Viewing the simulation visualization in this manner significantly enhances the user's understanding of the nets dynamic behavior, and is very useful in presentations to non-specialists and of course for functional verifying (debugging) the model behavior.

5. Conclusions

In this paper we have discussed the possibility of timed *GDPN* models to study the performance characteristics of hybrid systems. When dealing with real cases, it often happens that the *HPN* models tend to become very complex. In these cases *GDPN* models can be used as formal specification of system and automatically translated into detailed simulation programs in order to estimate performance result.

The construction and the analysis of complex timed *GDPN* models can only be done with the help of powerful automatic tools such as the user-friendly software *VHPN* tool whose performance modeling facilities and software architecture have been illustrated in this paper.

The integration within the same interface of the graphical facilities for the model construction, of the structural analysis algorithms for model validation, and of the control panels for performing the visual and interactive simulation, emphasizes the importance of including in a simulation experiment both validation and evaluation aspects of timed *GDPN* models.

The animation of the model, performed only when desired by the user, appears to be a powerful tool that complements the structural results for the

debugging and tuning of the models used for the performance analysis.

Already *VHPN* is used in educational process; it gives students a grasp of timed hybrid Petri net principles, introducing them into world for performance evaluation of computer systems and communication networks.

References

- [1] Alla, A., David, H. (1998) *Continuous and hybrid Petri nets*. Journal of Systems Circuits and Computers, 8(1), pp. 159-188.
- [2] Ajmone Marsan M., Balbo G., Conte G and Donatelli S. (1992) *Modelling with Generalized Stochastic Petri Nets*. Series in Parallel Computing. John Wiley & Sons.
- [3] David, R. (1997) *Modelling Hybrid Systems using Continuous and Hybrid Petri Nets*. In: Proc. of Intern. Conf. PNPM'97, pp.135-144.
- [4] Demongodin, I., Koussoulas, N. T. (1996) *Modelling Dynamic Systems Through Petri Nets*. Computational Engineering in Systems Applications, CESA'96 IMACS, IEEE-SMC, July 1996, Lille, France, pp. 279-284.
- [5] Demongodin, I., Koussoulas, N.T. (1998) *Differential Petri Nets: Representing continuous systems in a discrete-event world*, IEEE Transactions on Automatic Control, Vol. 43, No. 4.
- [6] Drath, R. (1998) *Hybrid object nets: An object oriented concept for modeling complex hybrid systems*. In.: Proc. Hybrid Dynamical Systems, 3rd International Conference on Automation of Mixed Processes, pp. 437-442.
- [7] Sanders W. S., Obal II W. D., Qureshi M. A. and Widjanarko F. W. (1995). *The UltraSAN Modeling Environment*. Performance Evaluation, 24(1), pp. 89-115.
- [8] Zimmermann A. and Freiheit J. (1998) *TimeNETs - An integrated modelling and performance evaluation tool for manufacturing systems*, in IEEE Int. Conf. On Systems, Man, and Cybernetics, San Diego, USA, pp. 535-540.