Journal of Engineering Science *Vol. XXX, no. 3 (2023), pp. 78 - 98*
Fascicle *Electronics and Computer Science* *ISSN 2587-3474*
Topic *Computers and Information Technology* *eISSN 2587-3482*

https://doi.org/10.52326/jes.utm.2023.30(3).06
UDC 004.032.26:519.7

SIMULATION AND ANALYSIS OF SPIKING NEURAL MEMBRANE COMPUTING MODELS BASED ON REWRITING TIMED HYBRID PETRI NETS WITH ANTI-TOKENS

Victor Moraru*, ORCID: 0000-0002-5454-8341,
Alexei Sclifos, ORCID: 0000-0003-4531-7944,
Emilia Sclifos, ORCID:0000-0003-1986-7256,
Emilian Guțuleac, ORCID: 0000-0001-6839-514X

Technical University of Moldova, 168 Stefan cel Mare Blvd., Chisinau, Republic of Moldova
*Corresponding author: Victor Moraru, *victor.moraru@calc.utm.md*

Received: 07. 16. 2023
Accepted: 08. 12. 2023

Abstract. In this paper we present the rewriting timed hybrid Petri nets (RTHPNs) enhanched with positive and negative place capacity, guard functions for transitions and rewriting rules, marking-dependent cardinality reversible arcs and anti-tokens. The RTHPN model allows its structure and/or attributes to change at run-time depending on its current state and/or the occurrence of some events. Also, we describe an approach to simulation and formal verification behaviour properties of *spiking neural membrane computing* (SNMC) models using particular RTHPNs that is supported by upgraded VPNP Tool. The use of RTHPNs in simulation and analysis of an extended SNMC model is illustrated through examples proving that such approach preserves faithfully its behaviours.

Keywords: *analysis, anti-token, spiking neural membrane computing, model, rewriting rules, hybrid timed Petri nets, simulation and verification.*

Rezumat: În lucrare prezentăm rețelele Petri hibride temporizate cu rescriere (RTHPNs) care sunt îmbunătățite cu capacități pozitive și negative ale locațiilor, funcții de gardă ale tranzițiilor și cele ale regulilor de rescriere, cu arce reversibile de cardinalitate marcaj - dependente și anti-tokene. Modelul RTHPN permite ca structura și/sau atributele sale să se schimbe în timpul rulării în funcție de starea curentă și/sau de apariția unor evenimente. Descriem și o abordare de simulare și verificare formală a proprietăților comportamentale ale modelelor de calcul membranal neuronal spiking (SNMC) folosind RTHPNs particulare, care sunt susținute de VPNP Tool actualizat. Folosirea RTHPN în simularea și analiza unui model SNMC extins este ilustrată în baza unui exemplu care demonstrează că o astfel de abordare păstrează fidel comportamentele acestuia.

Cuvinte-cheie: *analiză, anti-token, calcul membranal neuronal spiking, model, reguli de rescriere, rețele Petri hibride temporizate, simulare și verificare.*

1. Introduction

Spiking neural P systems (SNPS) belong to the third generation of neuronal models. They were proposed and studied in [1] as a class of distributed and parallel computing models that include the idea of spiking neurons into P systems [2, 3].

Next, due to the space restrictions, we will give a summary overview to this topic and refer the reader to papers [4 - 9] and the references therein. We only note that in recent years several variants of SNPS, with Turing computable sets of natural numbers, have been proposed by combining methods and ideas from the fields of biological activities, mathematics and computer science, accumulating rich results in their theoretical research with various applications [6, 7]. SNPS have a well defined network-distributed structure, a powerful parallel computing ability, dynamic characteristics and non-determinism. These characteristics of SNPS allow them to be applied in solving many practical problems [6].

Although great progress has been made in the field of theoretical definition of the application of different SNPS kinds in recent years they still have some shortcomings in distributed data processing, but they can be improved. Thus, traditional SNPS only processes integer numbers of spikes as symbolic data, so it is exceedingly difficult to process a large amount of numerical information with real values.

In [10] a new extension of SNPS is proposed, called *spiking neural membrane computing* (SNMC) models that improve on current SNPS variants by enabling real data processing technology. The SNMC model contains the *input data unit*, the *threshold unit* and evolution *rules* with a nonlinear time delay production function that are real or integer values. Synapse weights connecting neurons in SNMC models can have positive or negative values, and they transmit *spikes* or *anti - spikes*. Also, the Turing universality of the SNMC model is proved. Since the SNMC model can extend the application when information has integer and/or real values and this approach has great possibilities to solve some practical problems, for exemple, that are mentioned in [7, 8, 10].

In SNPS and SNMC, the firing rules are selected non-deterministically, so that any applicable firing rule is selected with equal probability. However this assumption is quite unnatural in what consists their application in different conditions. For this purpose, by introducing probabilities of selection of evolution rules in neurons, we propose an extension of the SNMS model, called ESNMS, similar to the formal framework proposed for SNPS [11] by using stochastic features.

Any developer of P systems and SNPS models and computing applications, knows that the most important quality of a computing application is that it is functionally correct, i.e. that it exhibits certain behavioral or *qualitative* properties [7, 8]. Once assured that the system behaves correctly, it is also important to ensure that the system meets also certain performance-related (or *quantitative*) objectives. Therefore, it is necessary that through well-constructed SNPS, SNMC or ESNMC models of the developed computational applications, the behavioral properties can be checked and thus possible errors that may appear in the earlier phases of the system development can be detected and corrected, since such an approach allows the modeler to fix them easily and cheaply.

Various aspects of the representations and qualitative characteristics of P and SNPS systems have been extensively studied. However, the tools that support simulation and behavior properties verification of real aplications using such models are relatively limited [12-14]. Most of the known SNPS simulators are mainly text-based that have little or no visualization of the models and their calculations. Several authors have proposed some approaches to simulate different variants of P systems [15-18] through appropriate Petri net (PN) extensions to remedy the mentioned disadvantages and analyze some behavioral properties of these models. PNs model is a graphical and mathematical modeling tool which is used to specify, in clear manner, the behaviors of concurrent systems. Moreover, due to the similarity of the graphical structure, the translations of SNPS into models of PNs and analysis of their behavior features, are carried out in [18]. For example, in [18] a variant of SNPS with spikes and anti-spikes are studied. It describes a method to represent and simulate SNPS with anti-spikes using PNs. With a well-constructed PN model of the elaborated SNPS or SNMC, faults in the system can be detected and fixed at earlier stages of development. PN simulation is a suitable and simple but effective approach for the modeller to verify the desired behavioral properties of discrete event systems. A list of PN simulation tools along with feature descriptions can be found on the Petri Nets Tools Database website [19].

However, to the best of our knowledge, with these tools we cannot visually simulate and analyze the behavioral properties of SNMC models using real data and stochastic application of evolution rules. Moreover, with the already known extensions of timed hybrid PN (THPN) [20] or fluid stochastic PN (FSPN) [21] it is very difficult to map SNMC models and analyze them, because in THPN places with negative-positive capacities and negative marking - dependent cardinality are not allowed. Also, it is easy to confirm from experience that the developed THPN or FSPN models, which adequately describe the behavior of real systems, are often difficult to use in practice due to the problem of the rapid increase in their graphic size. Thus, with a steady increase in complexity and size of SNMC, their models also become larger and less comprehen-sible. Introducing modularity concepts into system specifications is a wide range of research because it makes large descriptions handling easier.

The challenging development of large SNMC applications can be eased through the usage of appropriate models to simulate, evaluate and validate them before hand. One well known method for this is the deployment of hierarchical PNs (THiPN) that provide a more abstract view. Also, especially challenging is the development of large SNMC models with dynamic components [8] that allow dynamic structural adaptation. To overcome this problem, it is necessary to improve the THPN formalism that compactly and flexibly describe extended SNMC models with the probabilistic selection of evolution rules (ESNMC, for short).

Practical methodologies in engineering and computer science take a structural approach, designing systems from smaller subsystems and components, which can be combined and reused. In this context, for efficient formalization and to deal with the implementation and formal correctness analysis of SNMC and ESNMC models, in this paper we define a new extension of THPN, called rewriting THPN with anti-tokens (in short, RTHPN) having guards for transitions and rewriting rules. This approach allows to build modular and hierarchical models, capable of describing cases in which the structure of the model and its attributes can run-time change depending on its current state and/or the occurrence of some events. The RTHPN is the improved and extended version of the rewriting GSPN that are enriched with reconfigurability [22, 23].

As far as we know, there is no work in the literature that treats the visual simulation and anaysis of ESNMC models using dynamic run-time reconfiguration of RTHPN models. In this paper, we describe an approach that we believe is suitable in terms of both expressiveness and analysis capabilities of ESNMC models using RTHPNs. The proposed RTHPN models are based on the maximality step firing semantics [16].

Also, we present a methodology that maps SNMC and ESNMC models similarly into RTHPN representations, which allows visual simulation and the study of the behavioural properties of such models dynamics via the upgraded VPNP Tool [24] in an easy-to-use manner. The practical motivation is to propose a novel way to deal with the analysis complexity in some real-world applications under the framework of SNMC and ESNMC models. In this context, a numerical example is presented and studied to demonstrate the applicability and utility of the proposed RTHPN approach for simulation and analysis of ESNMC models. For this purposes the flat THPN (with anti-tokens) nets representation of the hierarchical RTHPN net can be used.

The paper is organized as follows. In Section 2 we describe the definition, spike evolution rules and behavior of ESNPC models. Also, in Section 3 we introduce the dynamic RTHPN with anti-tokens that allow the run-time reconfiguration of the analyzed ESNPC models. Section 4 provides the simulation and analysis metodology of ESNMC models using RTHPN nets. The conclusions of this work and future research efforts are described in Section 6.

2. Extended Spiking Neural Membrane Computing Models

In this section, based on SNMC model that is proposed in [10], an extended SNMC model, called ESNMC model, is presented. The definition and behavior of ESNMC models are given below. Neurons contain extended evolution rules with stochastic application with *integer* and/or *real* input value and a threshold value. The transmission of data (spikes or anti-spikes) by neurons is carried out by timed firing rules and their respective synaptic connections.

It is assumed that the readers are familiar with formal language theory and the basics of Membrane Computing (a good introduction is [2] with recent results and information in the P systems webpage [3]).

Definition 1**.** From [10] an ESNMC model of degree $m \geq 1$, denoted $G\Pi$, is a construct expressed by a 6-tuple, $G\Pi = (O, \Sigma, W, Syn, in, out)$, where:

(1) $O = \{a, \bar{a}\}$ is the binary alphabet, that a is called *spike* and $\bar{a}$ is called *anti-spike* included in neurons.

(2) $\Sigma = \{\sigma_1, \sigma_2, \cdots, \sigma_m\}$ is the set of neurons, of the form $\sigma_i = (u_i, b_i, pf_i, R_i)$, where: (*i*) $u_i \in R_i$ is input data in σ_i; (*ii*) $b_i \in R_i$ is a threshold of σ_i; (c) pf_i is the *production function* that compute the total data value of σ_i. The total value is the weighted sum of all inputs of σ_i minus the threshold; (*iii*) $R_i = \{r_{i,k_r}\}$ is a finite set of evolution rules of σ_i, with the form $r_{i,k_r} : (q_{i,k_r})E / a^{pf(u_i - b_i)_\varepsilon} \to a^{s_i}; \tau_{i,k_r}, \tau'_{i,k_r}$, $s_i \in \{0,1\}$, where: q_{i,k_r} is the application probability of evolution rule with $(\sum_{\forall k_r} q_{i,k_r}) = 1$; E is a regular expression over a or $\bar{a}$; and $\varepsilon = 0$ for integer values or $0 < \varepsilon < 0.5$ for real values of pf_i. The τ_{i,k_r} and τ'_{i,k_r} after the rule refers to time delay. The τ_{i,k_r} represent the time that neuron σ_i receive spikes from the $\sigma_l, l \neq i$, and τ'_{i,k_r} represents the rule execution time (from the execution of the production steps to the outputting step). If rule r_{i,k_r} is chosen non-deterministically, then it is not mentioned.

(3) $W = \{w_{i,j}, i \neq j\}$ is the weight on the synapse, which can be *positive* or *negative.* A *positive weight* generate *spikes*, and a *negative weight* generate *anti-spikes.*

(4) $Syn \subseteq \{1, 2, \cdots, m\} \times \{1, 2, \cdots, m\} \times W$ is the set of synapses.

(5) in and out are the input neuron and the output neuron, respectively. The input neuron converts the input data into spikes containing integer values or real values. The output neuron outputs the input data as a binary string composed of 0 *spikes* and 1 *spikes.*

A $G\Pi$ is pictorially described as a directed graph without self-loop, where the nodes of graph are represented by neurons, and the arcs indicate the synapses among the neurons, as shown in Figure 1. It also indicates the relationship between neurons. The set of arcs (synapses) entering in the neuron σ_i is denoted as ${}^{\bullet}\sigma_i$ and those that come out of σ_i are denoted as $\sigma_i^{\bullet}$.

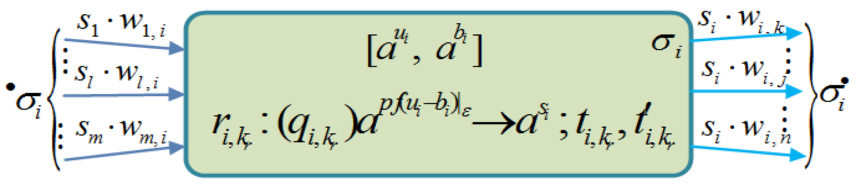


Figure 1. The neuronal structure of a neuron σ_i of $G\Pi$ model (Adapted from [10]).

Next, we explain the behavior of SNMC model. If threshold is 0, this means no threshold in neurons. The input data u_i of a neuron σ_i is the *original* data α_i plus *linking input* data $\sum_{\forall^{\bullet}\sigma_i}(s_l \cdot w_{l,i})$, namely $u_i = \alpha_i + \sum_{\forall\sigma_l}(s_l \cdot w_{l,i})$. The linking input data comes from the connected neurons, and the original data are that the σ_i itself already exists. For example, the real value 2.5 is shown as $a^{2.5}$ that represent as 2.5 spikes in a σ_i and $a^{-2.5}$ denot 2.5 spikes with a negative charge in the neuron, i.e. $2.5\bar{a}$. In each σ_i, an anti-spike can *immediately annihilate* one spike.

The $G\Pi$ model is *synchronized* by a global clock and works in a locally sequential at the level of each neuron and globally in maximal manner at whole $G\Pi$ model. In each σ_i, at firing step, if there is more than one rule enabled, then only one of them (chosen non-deterministically) can fire. At each step, the neurons of $G\Pi$ evolves in parallel and in a synchronising way, as all the neurons chooses an enabled rule and all of them fire at once.

The rules $R_i = \{r_{i,k_r}\}$ are processed as given below. So, the rule r_{i,k_r} contains two parts, including the *production function*, denoted pf_i, and the outputting of s_i. The pf_i is used to calculate $pf_i = u_i - b_i$, which will cause the state change of neuron σ_i. In addition, the neuron has a critical value, which is set to ε. Therefore, the execution steps of rules are divided into three steps: [10].

(1) *Production step*. When neuron σ_i receives weighted spikes s_l with data value $\sum_{\forall\sigma_l}(s_l \cdot w_{l,i})$ from connected $\sigma_l \in {}^{\bullet}\sigma_i$ neurons with σ_i at time τ_{i,k_r}, that represents the firing rule time τ_{i,k_r}. In σ_i is calculate the $pf_i = u_i - b_i$, where the $u_i = \alpha_i + \sum_{\forall\sigma_l}(s_l \cdot w_{l,i})$, α_i is the *original* data and b_i is the unchanged threshold value. Before a delay of t_{i,k_r} times, the σ_i is in a closed state.

(2) *Comparison step*. The result $pf_i = u_i - b_i$ is compared with the critical value ε, denoted $(u_i - b_i)|_\varepsilon$. It determines whether the output s_i of σ_i in the next step is 1 or 0.

(3) *Outputting step*. If $pf_i > \varepsilon$ then $s_i = 1$ and the rule r_{i,k_r} can be applied to output a spike with the value of 1. If it has $pf_i \le \varepsilon$, then $s_i = 0$ and the rule $(q_{i,k_r})E / a^{pf(u_i - b_i)_\varepsilon} \rightarrow \lambda; \tau_{i,k_r}, \tau'_{i,k_r}$ fires. Therefore, no spike can be sent by σ_i to the connected neurons with $\sigma_i^{\bullet}$. If r_{i,k_r} fire, the value unit u_i in σ_i is *consumed* (i. e. it is reset to 0) and the b_i is *unchanged*. The firing of r_{i,k_r} requires the following conditions: (1) Assume the number of spikes contained in neuron σ_i is γ, and a^{γ} belongs to the language set

represented by the regular expression E, and the current number of γ in σ_i is greater than or equal to the number of spikes consumed, u_i, i.e., $\gamma \geq u_i$. (2) The σ_i can only be activated when it receives the signal sent by the connected neurons ${}^{\bullet}\sigma_i$.

We mention that it can be shown that SNP systems are a particular case of the $G\Pi$.

As already shown, the neurons in an $G\Pi$ fire in parallel, that each neuron uses only one rule in each time unit τ. The current number of spikes (anti-spikes) present in each neuron of $G\Pi$ at that time is represented by the configuration, denoted as $C_k(\tau) = (u_1(\tau), u_2(\tau), \ldots, u_m(\tau))$. The initial configuration is denoted as $C_0(0) = (u_1(0), u_2(0), \ldots, u_m(0))$. Current $C_k(\tau)$ is changed by the locally sequential and globally maximal application of enabled rules. Such a step is called transition. The transition from $C_k(\tau)$ at time τ to the other $C'_k(\tau+1)$ configuration at time $\tau+1$ is denoted as $C_k(\tau)[\vartheta_\tau > C'_k(\tau+1)$, where $\vartheta_\tau \subseteq R$ is a set of executed enabled rules at time step τ. When the calculation reaches a certain configuration and there is no rule that can be activated, then the calculation stops, and this *halting configuration* is denoted as $C_h(\tau_H)$. The computational process of the $G\Pi$ can be regarded as a transition of a series of configurations, which is ordered and finite, i.e., from the initial configuration C_0 to $C_k(\tau)$.

Example of analysis of a $G\Pi$ model. Next, for the reader to better understand this approach, we will illustrate some definitions and behavior of $G\Pi$ models with an ESNMC model $G\Pi 1$ represented in Figure 2 that is modified from [10]. The $G\Pi 1$ has 5 neurons, denoted by σ_1, σ_2, σ_3, σ_4 and σ_5 that being the output one. The neurons are represented by nodes of a directed graph whose arcs represent the synapses; an arc also exits from the output neuron, pointing to the environment. In each neuron σ_i are specified the rules R_i, the threshold b_i and the u_i spikes present in the initial configuration $C_0(0)$.

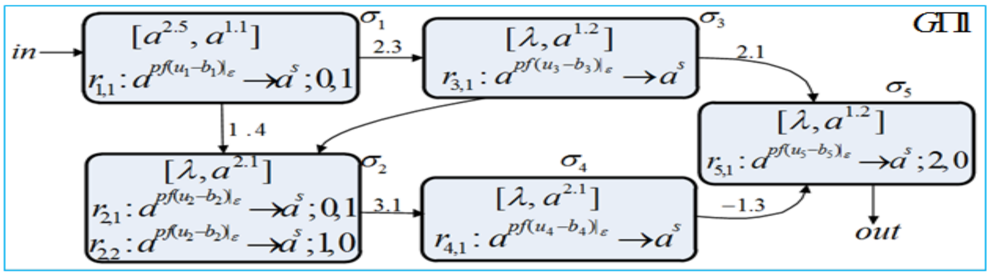

$G\Pi 1$
in
σ_1
$[a^{2.5}, a^{1.1}]$
$r_{1,1}: a^{pf(u_1-b_1)_\varepsilon} \to a^s; 0,1$
2.3
σ_3
$[\lambda, a^{1.2}]$
$r_{3,1}: a^{pf(u_3-b_3)_\varepsilon} \to a^s$
2.1
1 .4
σ_5
$[\lambda, a^{1.2}]$
$r_{5,1}: a^{pf(u_5-b_5)_\varepsilon} \to a^s; 2,0$
σ_2
$[\lambda, a^{2.1}]$
$r_{2,1}: a^{pf(u_2-b_2)_\varepsilon} \to a^s; 0,1$
$r_{2,2}: a^{pf(u_2-b_2)_\varepsilon} \to a^s; 1,0$
3.1
σ_4
$[\lambda, a^{2.1}]$
$r_{4,1}: a^{pf(u_4-b_4)_\varepsilon} \to a^s$
−1.3
out

Figure 2. An ESNMC model $G\Pi 1$ (Adapted from [10]).

The $G\Pi 1$ model works as follows. σ_1 contains a real input value $u_1 = 2.5$ spikes and a threshold value $b_1 = 1.1$, and it exists in the neuron in the form $[a^{2.5}, a^{1.1}]$. Let $\varepsilon = 0.1$ and time steps are denoted by $\tau_k = k, k > 0$ units of time. At time τ_1, neuron σ_1 fires so $pf_1 = (2.5 - 1.1) = 1.4 > \varepsilon$, and a spike $s_1 = 1$ is generated at time τ_2. Thus, σ_2 and σ_3 respectively receives together 1.4 spikes and 2.3 spikes from σ_1. At time τ_3, σ_3 generates $s_3 = 1$ because $pf_2 > \varepsilon$. Since, σ_2 contains two rules, of which one is selected for execution non – deterministically. Therefore, two cases can occur depending on the choice of rule in σ_2:

- If the $r_{2,1}$ is used, σ_2 receives a 1.4 spike from σ_1 at time τ_2, and at time τ_3 we get $pf_{2,1} = (1.4 - 2.1) < \varepsilon$. So, at time τ_4, σ_2 generate $s_2 = 0$. At the same time, σ_2 receives 1 spike from σ_3 and the $r_{2,2}$ is used. Since it's $pf_{2,2} < \varepsilon$, and σ_2 does not send a spike to σ_4. So the σ_4 has produces empty spikes at time τ_5, and σ_5 receives 2.1 spikes from σ_3. At time τ_6 , its rule in σ_5 fires and $pf_5 = (2.1-1) > \varepsilon$, so it produces $s_5 = 1$, and sends it out at the same time.

- If the $r_{2,2}$ is used then σ_2 is in the closed state before time τ_3 and does not receive any spikes. At time τ_3, the $pf_3 > \varepsilon$ produces spikes to send to neurons σ_2 and σ_5. Thus, at time τ_3, σ_2 receives 2.4 spikes: 1.4 from σ_1 and 1 from σ_3. At time τ_4, in σ_2 we get $pf_2 = (2.4 - 2.1) > \varepsilon$, it has $s_2 = 1$, and σ_2 produces $s_2 = 1$ and sends it to σ_4 then receives 3.1 spikes, and at time τ_5 its $pf_4 = (3.1 - 2.1) > \varepsilon$, so $s_4 = 1$. Hereupon, σ_5 receives 2.1 spikes from σ_3 and 1.3 anti - spikes from σ_4, so σ_5 contains $u_5 = (2.1 - 1.3) = 0.8$ spikes. In this way, at time τ_6 $s_5 = 1$and no sent out because $pf_5 = (0.8 - 1.2) < \varepsilon$.

The behaviour of the $G\Pi 1$ model can be analyzed based on a transition labelled directed graph of reachable configurations $C_i(\tau_k) = (u_1(\tau_k), u_2(\tau_k), \ldots, u_5(u_2(\tau_k))$ from the initial configuration C_0 that display the changes of spikes (anti - spikes) numbers $u_1(\tau_k)$ in σ_i at each time step τ_k by application of activated and executed set rules ϑ_{τ_k} in firing step.

Next, to deal with the visual simulation and formal correctness verification of ESNMC models $G\Pi$, we introduce a new extension of THPN, called Rewriting THPN (RTHPN), with negative place capacity; marking-dependent cardinality reversible arcs and the ability to dynamically in run-time reconfigure its structure and/or attributes.

3. Rewriting Timed Hybrid Petri Net with Anti-tokens

The definition of a RTHPN is derived according to [23, 25, 26] and inherits most of the HTPN [21], FSPN [22] and GSPN [23] characteristics. We assume that the readers are familiar with the basic concepts of these types of PN extensions. A more detailed theoretical description of these topics is beyond the scope of this article.

As already mentioned, this enhancement allows the compact modeling of high complexity SNPS and ESNMC models through RTHPN, without the risk of having a very complicated graphical size HTPN model, too difficult to represent and to understand.

Let IZ and IR be the sets of discrete and real numbers, respectively.

Definition 2. A Rewriting Timed Hybrid Petri Net (in short, RTHPN), denoted $RH\Gamma$, is a 16-tuple $RH\Gamma$ =<*P*, *E*, *Pre*, *Post*, *Test*, *Inh*, *K$_p$*, *K$_b$*, *Pri*, *G^E*, *G^R*, τ, ω, *V*, M_0, *Lib* >, where:

- *P* is the finite set of places partitioned into a set $P_d = \{p_1, \cdots, p_{n_d}\}$, $n_d = |P_d|$ of discrete places and a set $P_c = \{b_1, \cdots, b_{n_c}\}$, $n_c = |P_c|$ of continuous places (buffers), where $P = P_d \cup P_c$, $P_d \cap P_c = \varnothing$. 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;
- $E = T \cup \rho$ is a finite set of events, $T \cap \rho = \varnothing$, $P \cap E = \varnothing$, where T is a finite set of transitions and ρ is a finite set of discrete rewriting rules about the run-time structural and attributes change of $RH\Gamma$. The set E is partitioned into $E = E_0 \cup E_\tau$, $E_0 \cap E_\tau = \varnothing$ so that: E_τ is a set of timed events and E_0 is a set of immediate events. Likewise, E can be partitioned into a set $E_d = \{e_1, \cdots, e_{k_d}\}$, $k_d = |E_d|$ of discrete events and a set $E_c = T_c = \{u_1, \cdots, u_{k_c}\}$, $k_c = |E_c|$ of continuous transitions, where $T = T_d \cup T_c$, $T_d \cap T_c = \varnothing$. A transition $t_j \in T_d$ is drawn as a black

bar; a continuous transition $u_l \in T_c$ is drawn as an empty rectangle and rewriting rule $\hat{r}_k \in \rho$ is drawn as two embedded empty rectangles.

- *Pre*, *Test* and $Inh\!:\!P \times T \to Bag(P)$ respectively, are forward flow, test and inhibition functions with marking-dependent cardinality. *Pre* is the forward incidence function, *Test* is the promoter function and *Inh* is the inhibition function of transitions. $Bag(P)$ are discrete or real-valued multisets functions over *P* [20-22]. The backward flow function in the multisets of is $Post\!:\!T \times P \to Bag(P)$. These functions determine a mapping of the set of arcs A_{rcs} into set IZ of integer numbers (negative/positive) and set IR of real numbers which determines the marking-dependent cardinality of the arcs connecting the places (events) with the respective events (places). Also, the A_{rcs} set is partitioned into subsets:

$$A_{rcs} = A_d \cup A_h \cup A_t \cup A_c \cup A_s,\ A_d \cap A_h \cap A_t \cap A_c \cap A_s = \varnothing.$$

The subset A_d and A_s contains respectively the *discrete normal* and *continuous normal* set of arcs which can be seen as a function:

$A_d : ((P_d \times E) \cup (E \times P_d)) \times Bag(P) \to IZ$, and $A_s : ((P_c \times E) \cup (E \times P_c)) \times Bag(P) \to IR$.

The subsets of arcs A_d and A_s, are drawn as single arrows. The subset of discrete *inhibitory* and *test* arcs is $A_h, A_t : (P_d \times E) \times Bag(P) \to IZ$ or that of *continuous inhibitory* and *test* arcs is $A_h, A_t : (P_d \times E) \times Bag(P) \to IR$. These arcs are directed from a place to any kind event. The *inhibitory* arcs are drawn with a small circle at the end and *test* arcs are drawn as dotted single arrows. It does not consume the content of the source place. The subset A_c defines the *continuous flow* arcs $A_c : ((P_c \times T_c) \cup (T_c \times P_c)) \times Bag(P) \to IR$, and these arcs are drawn as double arrows to suggest a pipe. The arc of a net is drawn if the cardinality is not zero and it is labeled to the arc with a default value being 1;

- $K_p\!:\!P_d \to IZ$ is the capacity-function of discrete places and for each $p_i \in P_d$ this is represented by minimum capacity $K_{p_i}^{\min}$ and the maximum capacity $K_{p_i}^{\max}$, so that $-\infty < K_{p_i}^{\min} < K_{p_i}^{\max} < +\infty$, which can contain a discrete number of *tokens* (*anti - tokens*). By default, the $K_{p_i}^{\min} = 0$ and $K_{p_i}^{\max} \to +\infty$, and in this case no blocking effect occurs;
- $K_b : P_C \to IR$ is the capacity-function of continuous places and for each $b_i \in P_c$ 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, $x_i^{\min}$ = 0 and $x_i^{\max} \to +\infty$, and in this case no blocking effect occurs;
- *Pri*:$E \times Bag(P) \to IN_+$ defines the dynamic marking-dependent priority function for the *firing* of each *enabled event*, noted $e \in E(M)$. The firing of an enabled event with higher priority potentially disables all event $e \in E(M)$ with the lower priority;
- $G^E : E \times IN_+^{|P|} \to$ {*True*, *False*} is the set of *guard functions* associated with all event $e \in E$ and $G^R : \rho \times IN_+^{|P|} \to$ {*True*, *False*} is the set of *guard functions* associated with all *rewriting rule* $\hat{r} \in \rho$. For an $e \in E$ a guard function $g(e, M)$ will be evaluated in each current marking M, and if it evaluates to *True*, then event e may be enabled, otherwise e is disabled (by default it is *True*);
- $\tau : E_d \times Bag(p) \to IR^+$ is the firing delay time of respective discrete event $e_j \in E_d$, wherein the set E_d is partitioned into two subsets $E_d = E_0 \cup E_\tau$, $E_0 \cap E_\tau = \varnothing$, where E_0 is a set of *immediate* discrete events and E_τ is a set of *timed* discrete events, so that $\forall t_j \in T_0$ and

$\forall t_k \in T_\tau$, *Pri*(t_j)>*Pri*(t_k). The immediate transitions are drawn as black thin bars and have a zero firing delay time, i.e. for $t_j \in T_0$ the $\tau_j = 0$. The timed transitions are drawn as black rectangles and have a nonzero firing delay time, i.e. for $t_k \in T_\tau$ the $\tau_k > 0$;

- $\omega : T_0 \times Bag(P) \to IR^+$ is the weight function of immediate discrete transitions $t_k \in T_0$.

If an immediate transition $t \in T_0(M)$ is enabled in current (a vanishing) marking *M*, it fires with the following probability:

$$q(t, M) = \omega(t, M) / \sum_{t' \in T_0(M)} \omega(t', M);$$

- $V : T_c \times Bag(P) \to IR$ is the marking-dependent fluid rate function of timed continuous transitions $u_j \in T_c$. If u_j is enabled in *tangible* marking M it fires with rate $v_j(M)$, so that it continuously changes the fluid level of continuous places $\{b_k\} \in u_j^\bullet$.

- $M_0 =$ is the initial marking of $RH\Gamma$ net. The current marking (state) value of a $RH\Gamma$ net depends on the kind of place, and it is described by two vector-columns *M*=(***m***; ***x***), where ***m***: $P_d \to IZ$ and ***x***: $P_c \to IR$ are the marking functions of respective type of places. The discrete marking ***m*** $=(m_i, K_{p_i}^{\min} \le m_i \le K_{p_i}^{\max}, \quad \forall p_i \in P_d)$ with $m_i = \boldsymbol{m}(p_i)$ that describe the number of *tokens* in discrete place p_i, is represented by black dots (also allowed to take *negative* value, called *anti - tokens*). The marking ***x***= $(x_k, x_k^{\min} \le x_k \le x_k^{\max}, \forall b_k \in P_c)$ with x_k, which describes the fluid level in buffers b_k is a real number (real token), or *negative* real value (*anti - token*). The initial marking of net is $M_0 = (\boldsymbol{m}_0; \boldsymbol{x}_0)$. Vectors $\boldsymbol{m}_0$ and $\boldsymbol{x}_0$ give the marking of discrete places and of buffers, respectively.

- Lib is the set of $RH\Gamma_\kappa, \ \kappa = 1, 2 \cdots, n_\kappa$ subnet templates library involved in structural reconfiguration of the current $RH\Gamma$ by firing of enabled rewriting rules $\hat{r} \in \rho$.

Figure 3 shows the graphical representation of all $RH\Gamma$ primitives.

The role of the previous set of arcs 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 ***m***, i.e., the number of tokens (anti – tokens) in discrete place p_i when the marking is ***m***, (and x_k denote the *k*-th component of the vector ***x***, i.e. the fluid level in buffers b_k).

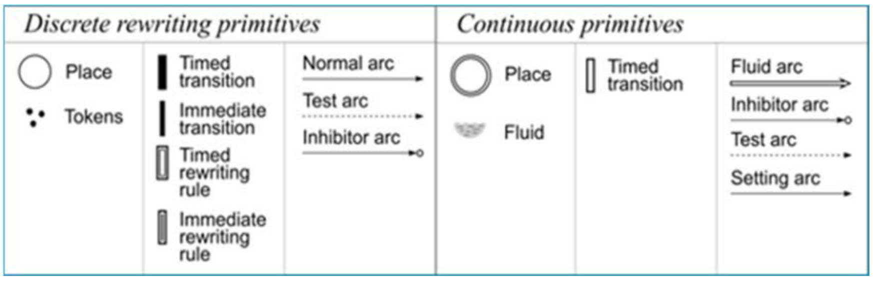


Figure 3. All primitives of a $RH\Gamma$ models.

Figure 4 shows the possible connections between discrete places and buffers with a discrete (resp. continuous) events through different types of arcs.

Enabling and firing rules of events. To define the $RH\Gamma$ enabling and firing rules, we introduce the following notations:

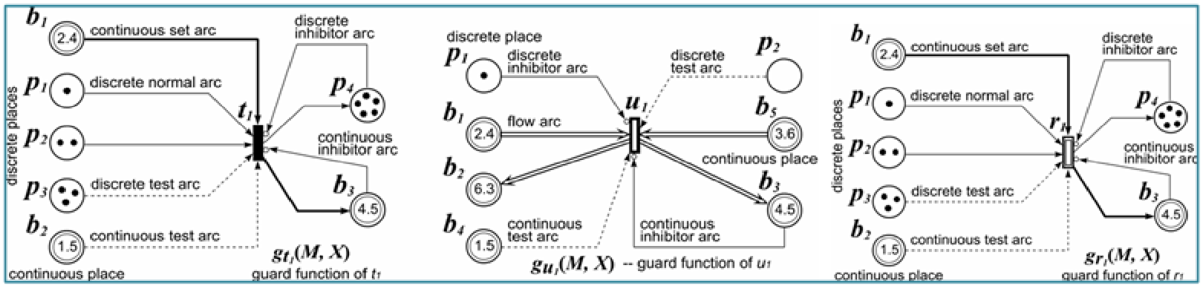


Figure 4. The posible connections between places with events through different types of arcs.

- $^{\bullet}e_j = \{p_i \in P / \Pr e(p_i, e_j) > 0\}$ the input set and $e_j^{\bullet} = \{p_i \in P / Post(e_j, p_i) > 0\}$ the output set places of event e_j, and with $^{\circ}e_j = \{p_i \in P / Inh(p_i, e_j) > 0\}$ the *inhibition* and $^{*}e_j = \{p_i \in P / Test(p_i, e_j) > 0\}$ are the *test* set places of transition t_j, respectively;
- *If* $(\Pr e(p_i, e_j) < 0)$ *then* $I_{j,i}^{-} = \Pr e(p_i, e_j)$; *If* $(\mathrm{Post}(e_j, p_i) < 0)$ *then* $O_{j,i}^{-} = \mathrm{Post}(e_j, p_i)$;
- *If* $(Inh(p_i, e_j) < 0)$ *then* $In_{j,i}^{-} = Inh(p_i, e_j)$; *If* $(Test(p_i, e_j) < 0)$ *then* $Ts_{j,i}^{-} = Test(p_i, e_j)$.

Also, we denote by $u_k \in T_c$ the continuous transitions and by $b_k \in P_c$ the buffers that can be distinct between discrete transitions and discrete places, respectively.

Let $E(M) = T(M) \cup \rho(M),\ T(M) \cap \rho(M) = \varnothing$ the set of *enabled events in current marking* M.

We say that an event $e_j \in E(M)$ is enabled in current marking M if the following logic expression (enabling condition $ec_j(M)$) is verified:

Definition 3. (*Enabling rule of events*) We say that an event e_j is enabled in current marking M, denoted $M[e_j >$, if the following *logic expression* $ec_j(M)$ is verified:

$$ec_j(M) = ec_j^{\Pr e}(M) \wedge ec_j^{Post}(M) \wedge ec_j^{Inh}(M) \wedge ec_j^{Test}(M) \wedge g_j(M)\text{, where:}$$

- $ec_j^{\Pr e}(M) = \bigwedge_{\forall p_i \in {}^{\bullet}e_j} ((M(p_i^e) \geq \mathrm{Pre}(p_i, e_j)) \wedge ((K_{p_i}^{\max} - M(p_i)) \geq -I_{j,i}^{-}))$ is the *enabling condition* relative to the normal arcs input, that are incident to e_j and to the capacities of the places $p_i \in {}^{\bullet}e_j$, and $M(p_i^e) = M(p_i) - K_{p_i}^{\min}$ is the effective number (discrete or real) of tokens in p_i;
- $ec_j^{Post}(M) = \bigwedge_{\forall p_i \in e_j^{\bullet}} ((M(p_i^e) \geq Post(e_j, p_i)) \wedge ((K_{p_i}^{\max} - M(p_i)) \geq -O_{j,i}^{-}))$ is the *enabling condition* relative to output normal arcs, that link these incident places to the transition e_j and to the place capacities of $p_i \in e_j^{\bullet}$;
- $ec_j^{Inh}(M) = \bigwedge_{\forall p_i \in {}^{\circ}e_j} ((M(p_i^e) < Inh(p_i, e_j)) \wedge (-M(p_i^e) < In_{j,i}^{-}))$ is the *enabling condition* relative to *inhibitory arcs*;
- $ec_j^{Test}(M) = \bigwedge_{\forall p_i \in {}^{*}e_j} ((M(p_i^e) \geq Test(p_i, e\ )) \wedge (M(p_i^e) \geq -Ts_{j,i}^{-}))$ is the *enabling condition* relative to inhibitory *test arcs*.

Also, we note *by* $E_d(M) = T_d(M) \cup \rho(M), T(M) \cap \rho(M) = \varnothing$ the set of enabled discrete events in a current marking *M*, where $T_d(M)$ and $\rho(M)$ are the sets of enabled discrete transitions and enabled rewriting rules, respectively.

In $RH\Gamma$ models, concurrency of enabled events is also represented in a natural way. Two or more enabled events are concurrent at a given marking M if they can be fired at the same time, i.e. simultaneously. Every event enabled by a current marking M can fire but is never forced to fire if they are in conflict. Conflict occurs between events that are enabled by the same marking, where the firing of one event disables the other. In this case, it is necessary to define how and when a certain conflict should be resolved, which leads to non-determinism of its behavior.

Firing rules of enabled events. Let ${}^{A}W = \{{}^{\bullet}W,\ W^{\bullet},\ {}^{\circ}W\}$ be the weights of the respective type arcs in A_{rcs} and $E_d \subset E$ is the set of discrete events. We note $Atr_\Gamma = \{{}^{A}W,\ K_p, K_b, Pri, G^E, G^R, \tau, \omega, V\}$ the set of *quantitative attributes* of the currently activated (sub)nets type $RH\Gamma$. Also, let $RN = <\Gamma, M>$ be the current configuration of $RH\Gamma$, where $\Gamma = RH\Gamma \setminus M$ and M is the current marking of $RH\Gamma$.

A dynamic reconfiguration of current *RN* by the firing of enabled $\hat{r} \in \rho$ is a map $\hat{r}: \{RHL, Atr_L\} \triangleright \{RHR, Atr_R\}$, where $\{RHL \in Lib_{RN},\ Atr_L \in Lib_{Atr}\}$ is the *left-hand side* and $\{RHR \in Lib_{RN},\ Atr_R \in Lib_{Atr}\}$ is the *right-hand side* of the *rewriting operator* $\triangleright$ assigned to rewriting rule $\hat{r}$, respectively. The $\triangleright$ represents a *binary rewriting operation* which produces a *structural change* and/or *change of attributes* in *RN* by replacing (rewriting) the fixed current subnet $\{RHL, Atr_L\} \subseteq RN$, RHL are dissolved (with $P_L \subseteq P$, $E_L \subseteq E_d$ and subset of arcs $A_L \subseteq A$ and/or Atr_L are deleted) and in run-time, a new $\{RHR, Atr_R\}$ subnet (with $P_w \subseteq P$, $E_w \subseteq E_d$ and set of arcs A_w and/or Atr_R are added) belong to the new modified resulting underlying net $RN' = (RN \setminus RHL) \cup RHR$ with $P' = (P \setminus P_L) \cup P_w$ and $E' = (E_d \setminus E_L) \cup E_w$, $A' = (A - A_L) + A_w$ where the meaning of \ (and $\cup$) is operation of removing (adding) *RHL* from (RHR to) current *RN*. In this new RN' net, obtained by the firing of $\hat{r} \in \rho(M)$, the same elements (places, events and arcs with respective specified attributes) belonging to RN' are respectively *merged* [24]. For example, when merging the same place $y \in P$ from two different RN_i and RN_j subnets with the respective current marking $m(y) = n_i$ and $m(y) = n_j$ in this place, the resulting number of tokens (anti-tokens) in this place will add up: $m(y) = (n_i + n_j)$.

The current *tangible state configuration* of a RN net is $\gamma_\tau = (\Gamma, M)$, i.e. the current structure configuration of the Γ net together with a current marking *M* at time τ . Also, the *tangible* $\gamma_0 = (\Gamma_0, M_0)$ is the initial configuration of analyzed $RH\Gamma$.

Firing rule of enabled discrete events. An enabled event $e_j \in E_d(M)$ fires if *no other event* $e_k \in E_d(M)$ with higher priority has been enabled. Hence, for each $e_j \in E_d(M)$ ***if*** $((e_j = t_j) \vee (e_j = \rho_j) \wedge (g^R(\hat{r}_j, M) := "False"))$ ***then*** (the firing of $t_j \in T_d(M)$ or firing of $\rho_j \in \rho(M)$ changes only the current marking of *RN*: $((\Gamma, M)[e_j > (\Gamma, M')) \Leftrightarrow (\Gamma = \Gamma$ and $M[e_j > M'$ in Γ)), where $M' = M - \mathrm{Pr}\,e(\vartheta, \cdot) + Post(\vartheta, \cdot)$. The vectors $\mathrm{Pr}\,e(\vartheta, \cdot)$ and $Post(\vartheta, \cdot)$ are the functions induced by those respective $\mathrm{Pr}\,e$ and $Post$ incidence matrices of the $RH\Gamma$ [21-24]. Also, for every $e_j \in E_d(M)$, ***if*** $((e_j = \rho_j) \wedge (g^R(\rho_j, M) := "True"))$ ***then*** the event e_j occurs at firing of the rewriting rule $\rho_j \in \rho(M)$ and it changes the configuration $\gamma_\tau = (\Gamma, M)$ and marking of the current *RN* net, so that: $((\Gamma,\ M)[\rho_j > (\Gamma,\ M')) \Leftrightarrow (\Gamma = \Gamma'$ and $M[\rho_j > M'$ in Γ'), i. e. $\gamma_{\tau'} = (\Gamma', M')$.

The firing of an *immediate* discrete event $e_j \in E_d(M)$ enabled in marking $M =$ (***m***, ***x***) yields a new *vanishing marking* $M' =$ (***m'***, ***x***). We can write (***m***, ***x***) [e_j > (***m***, ***x'***). If the marking *M* = (***m***, ***x***) is *tangible*, the fluid could continuously flow through the flow arcs A_c of enabled continuous transitions into or out of continuous places (buffers) $b \in P_c$. As a consequence, a continuous transition $u \in T_c$ is *enabled* at current marking M if for every buffer, fluid level increases or decreases, and its *enabling degree* is: $enab(u, M) = \min_{b \in {}^{\bullet}u}$ {$\boldsymbol{x}(b)/Pre(u, b)$} [21]. The reachability state graph (in short, $RG_{RH\Gamma}$) of configurations $\gamma_\tau = (\Gamma, M)$ from initial configuration $\gamma_0 = <\Gamma_0, M_0>$ is the *labeled directed graph* whose nodes are the states γ_τ and whose arcs, which are labeled with fired events $\theta_\tau \in E_d(M)$ of RN , are of two kinds:

(*i*) $(R\Gamma, M)[e_j > (R\Gamma, M')$ if $((e_j = t_j) \vee (e_j = \hat{r}_j)) \wedge (g^R(\rho_j, M) := "False")$;

(*ii*) $(R\Gamma, M)[\rho_j > (R\Gamma', M')$ if $(e_j = \rho_j) \wedge (g^R(\rho_j, M) := "True")$.

This enhancement allows the compact representation of high complexity analyzed ESNMC models, without the risk of having a very complicated and too difficult to understand graphical presentation through RTHPN model. Also, $RH\Gamma$ model supports a definition of a modular and hierarchical design methodology.

We note the fact that in any configuration $\gamma_\tau = (\Gamma, M)$, *tokens and anti-tokens cannot coexist* in the same place of $RH\Gamma$, they will *immediately annihilate* each other, so the neurons always contain either only spikes or anti-spikes. The annihilation is possible in each marking M with $m_i = M(p_i) > 0$, (resp. $x_i = M(b_i) > 0$)), *tokens* and $a_i = M(p_i) < 0$, (resp. $y_i = M(b_i) < 0$), *anti-tokens*, in the same place. This mutual annihilation of spikes and anti-spikes takes no time. This action produces a *vanishing state* that immediately changes the couple (m_i, a_i), (resp. (x_i, y_i)). More precisely, both m_i and a_i, (respective x_i and y_i), will decrement simultaneously when the values are non-zero. This implies that in each place p_i (buffer b_i) we can have a current marking m_i either with $0 \le m_i \le K_{p_i}^{\max}$, (respectively $0 \le x_i \le K_{b_i}^{\max}$) or with a_i , $K_{p_i}^{\min} \le a_i < 0$ (respectively $K_{b_i}^{\min} \le y_i < 0$). As a result, if ($\mathrm{Pre}(p_i, e_j) > 0$) then the firing of event $e_j \in E(M)$ consumes from (produces in) the same place p_i (buffer b_i), a number $\mathrm{Pre}(p_i, e_j) > 0$ of tokens (anti-tokens). Otherwise, it produces in (consumes from) the same place a number $\mathrm{Pre}(p_i, e_j) < 0$ of anti-tokens (tokens). Also, if ($\mathrm{Post}(e_j, p_i) > 0$) then the firing of event $e_j \in E(M)$ produces in (consumes from) the place p_i (buffer b_i) a number of tokens (anti-tokens), otherwise it consumes from (produces in) the same place a number $\mathrm{Post}(e_j, p_i) < 0$ of tokens (anti-tokens).

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

We allow the firing delay τ_j and the enabling functions of the timed discrete events $e_j \in E_d(M)$, the firing speeds v_k and enabling functions of the timed continuous transitions $u_k \in E_c(M)$ and arc cardinalities (positive or negative values) to be dependent on the current state of the $RH\Gamma$, as defined by the current marking *M*.

Next, for accurate translating $G\Pi$ models into $RH\Gamma$ models, we consider $RH\Gamma$ models only with *firing step semantics of discrete events* $E_d \neq \varnothing$ and in which $T_c = \varnothing$. If in a $RH\Gamma$ model the set of rewriting rules is empty, i.e. $\rho = \varnothing$, then we deal with particular flat model, denoted as $H\Gamma$.

An example of a $H\Gamma 1$ subnet with negative capacities of places and arcs, whose marking-dependent cardinality can be negative, is shown in Figure 5a. In $H\Gamma 1$ the minimum capacity of the places are $K_{p_i}^{\min} = K_{b_i}^{\min} = -5$, *i*=1,...,4 and the arcs with weights that have negative values are $Post(t_1, p_4) = -3$, $Post(t_1, b_3) = -4.5$, and the others have positive weights.

The initial marking of the subnet $H\Gamma 1$ in Figure 5a is $M_0 = (3,\ 0,\ 4,\ 5; 4.5, 3.8, 1.5, -2.7)$. The firing of transition t_1 produces a new marking $M_1 = (1,\ 5,\ 3,\ 2; 2.0, 2.8, -3.0, -1.7)$, i.e. $M_0[t_1 > M_1$. This fact is shown in Figure 5b. We note that the *negative weight* of some arcs leads to a change in the direction of the tokens flow, thus modeling *reversible arcs*.

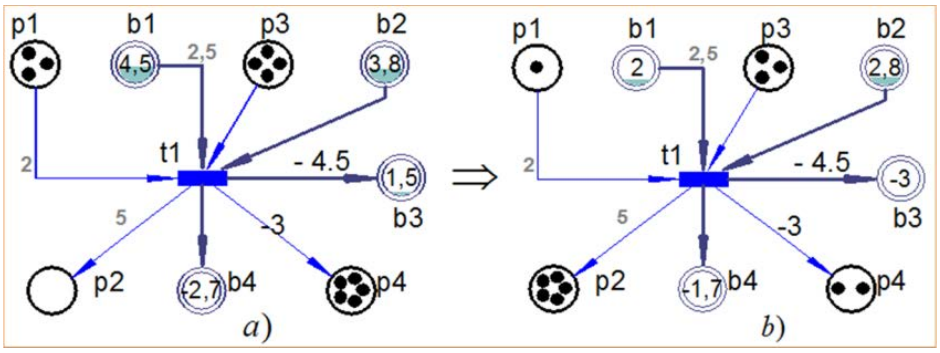


Figure 5. Discrete transition t_1 firing of a $H\Gamma 1$ flat subnet.

Parallel activities can be easily expressed in terms of $RH\Gamma$ using maximally firing *step* semantics in which executions of enabled discrete events $E_d \subset E$ are represented by a sequence of steps [15 – 18], [20]. Steps in $RH\Gamma$ are sets of enabled discrete evens that fire independently and parallel at the same time, i.e., simultaneously. The change in the marking and/or configuration of the $RH\Gamma$ when a step occurs is given by the sum of all the changes that occur for each event.

Definition 4. *(Firing step)* A *firing step* in $RH\Gamma$ at time τ is a set θ_τ of enabled events which are free enabled in a current configuration $\gamma_\tau = (\Gamma, M)$, i.e. $\theta_\tau \subset E(\Gamma, M)$. A firing step θ_τ can be executed at time τ_k leading to the new marking M' in $\gamma'_{\tau_k} = (\Gamma, M')$ or new configuration $\gamma''_{\tau_k} = (\Gamma', M')$ if there is no other $\theta'_\tau \subset \gamma_\tau$ enabled set of events with a higher priority than θ_τ.

A computation of a $RH\Gamma$ is a finite or infinite sequence of step executions at time τ_k starting from the initial configuration $\gamma_0 = (\Gamma_0, M_0)$ and every configuration $\gamma_{\tau_k} = (\Gamma_k, M_k)$ appea-ring in such a sequence is called reachable.

We note that the modeling power of $RH\Gamma$ *nets* is equal to the Turing machine, because it contains guard functions and/or inhibitory arcs [19, 26]. In [28] it was demonstrated that for any Petri net model with *reset arcs* the accessibility property is not decidable. In general, the accessibility properties of $RH\Gamma$ are not decidable, because in this type of Petri net the reset arcs can be described through marking-dependent functions of arcs. But, for particular cases of $RH\Gamma$, some behavior properties can be *decidable*.

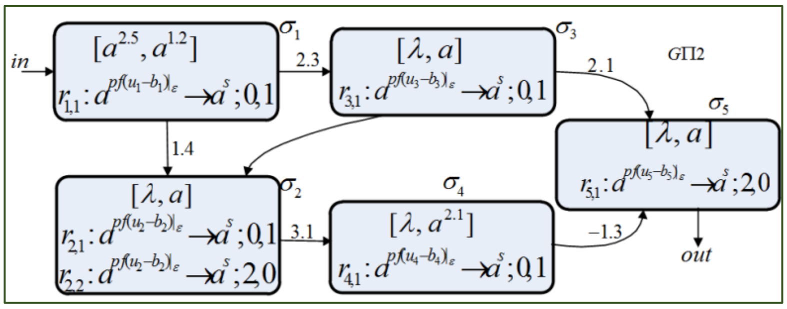
4. Simulation and analysis of ESNMC models using $RH\Gamma$ nets

The behaviour of the SNPS and ESNPC models is often similar to TPN [17, 18] and also of the one $RH\Gamma$ nets, where is used a time maximally firing step semantic of enabled events. So, a major strength of $RH\Gamma$ is their support for analysis of many behavioural properties associated with SNPS [17, 18], ESNPC models and $RH\Gamma$ such as *reachability*, *boundedness and safeness*, *liveness, terminating, deadlock–free* [19-22, 24]. We are checking these properties for

SNMC and ESNMC models based on $RH\Gamma$, that provide relevant information about the behavioral properties of the systems.

Next, we will describe an approach to translate SNMC and ESNMC models into $RH\Gamma$ nets with step semantics for the visual simulation and to study the behavioural properties of such models. For this one, we illustrate the applicability of our approach by visual simulation and analyze the behavioural properties a SNMC and ESNMC models using VPNP Tool [24].

First, to give the pictorial visibility of RTHPN models, similar to that of SNMC and ESNMC models, we will first show how a $RH\Gamma 0$ model describing the behavior of a neuron σ_i of a $G\Pi$ model is constructed. Then, we will show how the model $RH\Gamma 1$ is constructed, which adequately describes the behavior of the $G\Pi 2$ model that is presented in Figure 6.\



in
$[a^{2.5}, a^{1.2}]$
$r_{1,1}: a^{pf(u_1-b_1)_\varepsilon} \xrightarrow{s} a; 0,1$
σ_1
2.3
$[\lambda, a]$
$r_{3,1}: a^{pf(u_3-b_3)_\varepsilon} \xrightarrow{s} a; 0,1$
σ_3
2.1
$G\Pi 2$
σ_5
1.4
$[\lambda, a]$
$r_{5,1}: a^{pf(u_5-b_5)_\varepsilon} \xrightarrow{s} a; 2,0$
σ_2
σ_4
$[\lambda, a]$
$r_{2,1}: a^{pf(u_2-b_2)_\varepsilon} \xrightarrow{s} a; 0,1$
$r_{2,2}: a^{pf(u_2-b_2)_\varepsilon} \xrightarrow{s} a; 2,0$
3.1
$[\lambda, a^{2.1}]$
$r_{4,1}: a^{pf(u_4-b_4)_\varepsilon} \xrightarrow{s} a; 0,1$
−1.3
out

Figure 6. The ESNMC model $G\Pi 2$ (Adapted from from [10]).

Figure 7 presents an ESNMC2 model $RH\Gamma 0$ that translates neuron σ_i of a $G\Pi$ model, which is shown in Figure 1. There the neuron σ_i is represented using a continuous place $b_{i,1}$, which is connected to the timed rewrite rule $\rho_{i,1}$ by a test arc with weight $x_{i,1}$ that is the current number of spikes in buffer $b_{i,1}$. In $RH\Gamma$ models, places (transitions) correspond to local states (events, actions, activities) and rewriting rules can modify, reconfigure these models. The arc between a place and transitions (rewrite rule) and places represents axons.

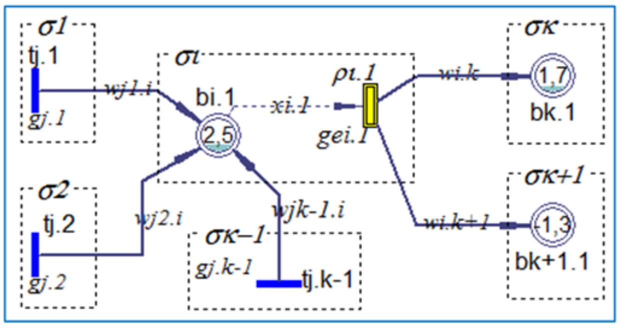


Figure 7. A $RH\Gamma 0$ model of neuron σ_i in $G\Pi$ model that is showed in Figure 1.

The attribute notations in all the following figures, that represent $RH\Gamma$ or $H\Gamma$ flat models, will be interpreted in the following way: $yi.j := y_{i,k}$, where $y \in \{b, p, t, \rho, g, ge, w, x\}$ and the first index i shows the order number of the neuron σ_i, and the second index k shows the order number of the symbol (attribute) y within this neuron or this $G\Pi$ model.

The meanings of the places, transitions and rewriting rules of the $RNR\sigma_i$ *model are*:

- **Continuous places**: in $b_{i,1}, b_{k,1}$ and $b_{k+1,1}$ is stored (memorized) the respective current numbers $x_{i,1}, x_{k,1}$ and $x_{k+1,1}$ of spikes (anti- spikes).
- **Immediate transitions**: the firing of $t_{j,1}, t_{j,2}$ and $t_{j,k-1}$ transmit a respective number (equal to respective weight) $w_{j1,i}, w_{j2,i}$ and $w_{jk-1,i}$ of spikes (anti- spikes) in continous places $b_{i,1}$.

• **Timed rewriting rule**: at firing of rewriting rules $\rho_{i,1}:\{\rho_{i,1}\}\triangleright RNR\sigma_i$ the $\rho_{i,1}$ is *removed* with its incident arcs and they are simultaneously replaced with the flat subnet $RNR\sigma_i \in Lib$ that is shown in Figure 8 In this case the places $b_{i,1}$, $b_{k,1}$ and $b_{k+1,1}$ are respectively merged, so that the number of spikes (anti - spikes) in respective place of $RH\Gamma 0 \setminus \{\rho_{i,1}\}$ and $RNR\sigma_i$ will be equal to their sum.

The guard function of $\rho_{i,1}$ which determines the enabling condition is $ge_{i,1} := (x_{i,1} \neq 0)$, and the one which determines the firing condition is $gr_{i,1} :=" True"$.

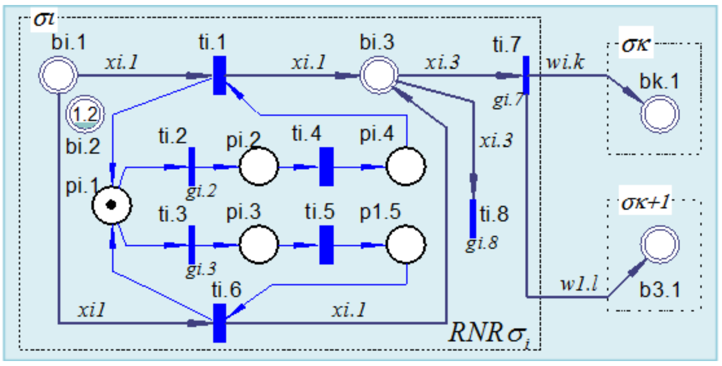


Figure 8. Flat $RNR\sigma_i$ flat subnet which substitutes $\rho_{i,1}$ in $RH\Gamma 0$ model shown in Figure 7.

The initial marking of $b_{i,1}$, $b_{k,1}$ and $b_{k+1,1}$ in $RH\Gamma 0$ is $M_{RH\Gamma 2} = (\mathbf{0};\mathbf{x}) = (\mathbf{0}; 2.5, 1.7, -1.3)$. At the firing of $\rho_{i,1}$, a respective number of spikes (anti- spikes), equal to the respective weights $w_{i,k}$ and $w_{i,k+1}$, will be transmitted in respective place $b_{k,1}$ and $b_{k+1,1}$.

The meanings of the places and transitions of the model $RNR\sigma_i$ subnet, with 2 spike evolution rules $\rho_{i,1}$ and $\rho_{i,2}$ of σ_i *are*:

• **Continuous places**: in $b_{i,1}$, $b_{k,1}$ and $b_{k+1,1}$ are stored (memorized) the respective current numbers $x_{i,1}$, $x_{k,1}$ and $x_{k+1,1}$ of spikes (anti- spikes). These places will merged respectively with those in the $RH\Gamma 2$ model; in $b_{i,2}$ is stored the threshold value; $b_{i,3}$ - buffer to check the value of production function $pf_i = (x_{i,3} - x_{i,2})|_\varepsilon$ with critical value ε.

• **Discrete places**: $p_{i,1}$ - initializing the probabilistic selection of a single evolution rule $\rho_{i,1}$ or $\rho_{i,2}$ of σ_i; $p_{i,2}$ (resp. $p_{i,3}$) - setting the time of $\rho_{i,1}$ (resp. $\rho_{i,2}$) that neuron σ_i receives spikes from the ${}^{\bullet}\sigma_i$; $p_{i,4}$ (resp. $p_{i,4}$) - setting the rule $\rho_{i,1}$ (resp. $\rho_{i,2}$) execution time.

•**Timed discrete transitions:** $t_{i,1}$ (resp. $t_{i,6}$) - the execution time $\tau_{i,1}$ (resp. $\tau_{i,2}$) of rule $\rho_{i,1}$ (resp. $\rho_{i,2}$); $t_{i,4}$ (resp. $t_{i,5}$) - the time $\tau'_{i,1}$ (resp. $\tau'_{i,2}$) of rule $\rho_{i,1}$ (resp. $\rho_{i,2}$), that neuron σ_i receive spikes from the ${}^{\bullet}\sigma_i$.

• **Immediate discrete transitions**: $t_{i,2}$ (resp. $t_{i,3}$) - mutual exclusion activation of rule $r_{i,1}$ (resp. $r_{i,2}$). These transitions perform the probabilistic selection of evolution rules. By default, selection is performed non-deterministically; $t_{i,7}$ - generation of spike s_i, guard function $g_{i,7} := (pf_i > \varepsilon)$, and transmision of spike $s_i \cdot w_{i,k}$ to neurons $\sigma_k^{\bullet}$; $t_{i,8}$ - no spikes can be sent to the connected neurons with $\sigma_i^{\bullet}$, guard function $g_{i,8} := (pf_i \leq \varepsilon)$, the value unit $u_i = x_{i,3}$ of buffer $x_{i,3}$ is *consumed*.

The guard function of $t_{i,2}$ and $t_{i,3}$ is $g_{i,2} := (m_{i,1} \neq 0)$ and $g_{i,3} = g_{i,2}$, respectively. As a result, at firing of $\rho_{i,1}$ rewriting rules of $RH\Gamma0$, will reconfigure itself and we get the flat $H\Gamma1$ model that is represented in Figure 9.

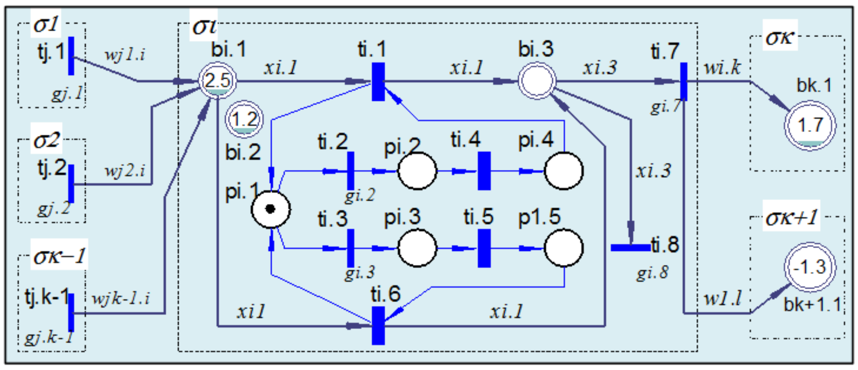


Figure 9. A flat $H\Gamma1$ model of an neuron σ_i with 2 rules.

If it is necessary to analyze the behavioral properties of an ESNMC model, which contains several $G\Pi$ type neurons, it can be mapped into a $RH\Gamma$ type model, using the approach used to build the model $RH\Gamma0$ of neuron σ_i. In this context, we will consider as an example the $G\Pi2$ model, shown in Figure 6, for which we construct the $RH\Gamma1$ model (see Figure 10) whose running behavior is equivalent to the $G\Pi1$ model. We notice that there is a similarity between the structures of these two types of models. The reader might notice that the structure of the $RH\Gamma2$ model is similar to the one $G\Pi2$ from Figure 6.

The meanings of the places, transitions and rewriting rules of the $RH\Gamma1$ *model are*:

• **Continuous places**: in $b_{i,1}$, $i = 1, 2, \ldots, 5$ is stored (memorized) the respective current numbers $x_{i,1}$, $i = 1, 2, \ldots, 5$ of spikes (anti- spikes) in neuron σ_i; $b_{0,1}$ - the environment receives the transmitted spikes;

• **Timed rewriting rule**: at firing of rewriting rules $\rho_{i,1} : \{\rho_{i,1}\} \triangleright RHRi.1$ the $\rho_{i,1}$, $i = 1, \ldots, 5$ and $\rho_{i,0} : \{\rho_{i,0}\} \triangleright RHR0$ are *removed* with its incident arcs and they are simultaneously replaced with the respective template subnets $\{RHRi.1, RHR0\} \in Lib$ that are shown in Figure 11. In this case the places $b_{i,1}$ are respectively merged, so that the number of spikes (anti - spikes) in respective place of $RH\Gamma1 \setminus \{\rho_{i,1}\}$ and template subnets will be equal to their sum.

The guard function of $\rho_{i,1}$, $i = 1, 2, \ldots, 5$ which determines the enabling condition is $gei.i = g^E_{i,1} := (x_{i,1} \neq 0)$, and the one which determines the firing rewriting condition, by default is $g^R_{i,1} :=$"$True$". The initial tangible marking of $b_{i,1}$, $i = 1, 2, \ldots, 5$ in $RH\Gamma1$ is $\mathbf{x} = (2.5, 0, 0, 0, 0) = (2.5b_{1,1})$. At the firing of $\rho_{i,1}$, a respective number of spikes (anti- spikes), equal to the respective arc weights $w_{i,k}$ and $w_{i,k+1}$, will be transmitted in respective place $b_{i,1}$, $i = 1, 2, \ldots, 5$ and $b_{0,1}$.

In the result of run-time reconfiguration of $RH\Gamma1$ model will be obtained the flat $H\Gamma2$ model that is shown in Figure 12. The meanings of the respective places, transitions, arc weights and guard function of the $H\Gamma2$ model are the same as in the $H\Gamma1$ model that is shown in Figure 9. In case that in $RH\Gamma$ models the input value of the neurons σ_i can be positive (negative) integer numbers then continuous places $b_{i,k}$, $k = 1, 2, 3$, will change to respective discrete places

$p_{i,k}$, $k = 1, 2, 3$. Likewise, if in $RH\Gamma$ models the execution time $\tau_{i,k_r} = 0$ and/or the time $\tau'_{i,k_r} = 0$, neuron σ_i receives spikes from the ${}^\bullet\sigma_i$ then the respective *timed* discrete transitions will change to respective *immediate* discrete transitions (as example, see $t_{2,2}$ in $RHR2.1$ and $t_{5,1}$ in $RHR5.1$ subnet) or they are removed (see $RHR1.1$, $RHR3.1$ and $RHR4.1$ subnets).

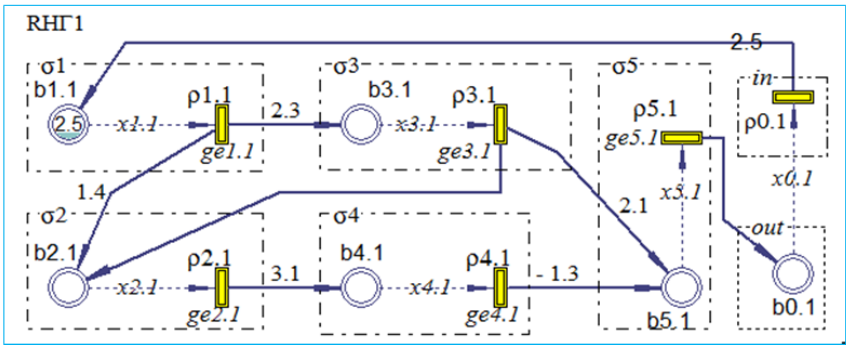


Figure 10. The $RH\Gamma1$ model of $G\Pi1$ model that is shown in Figure 6.

In order to perform the visual simulation and check the behavioral properties of the $H\Gamma$ flat models we have developed and integrated into VPNP Tool [24] a special software module. The Graphical User Interface (GUI) allows an intuitive, user-friendly tool for creating and editing $H\Gamma$ nets in an easy, fast and effcient way. Users are able to perform tasks using a menu bar, a toolbar and mouse actions. The $H\Gamma$ nets can be printed or exported into graphical format bitmap. VPNP Tool supports the parallel execution of events. It offers a visual animator so that the user can in step-by-step mode or automatically experiment with the token game, firing any of the enabled events (transitions and rewriting rules) at each firing step. Animation history is recorded, i.e. all the time fired events can be seen on the side of the screen.

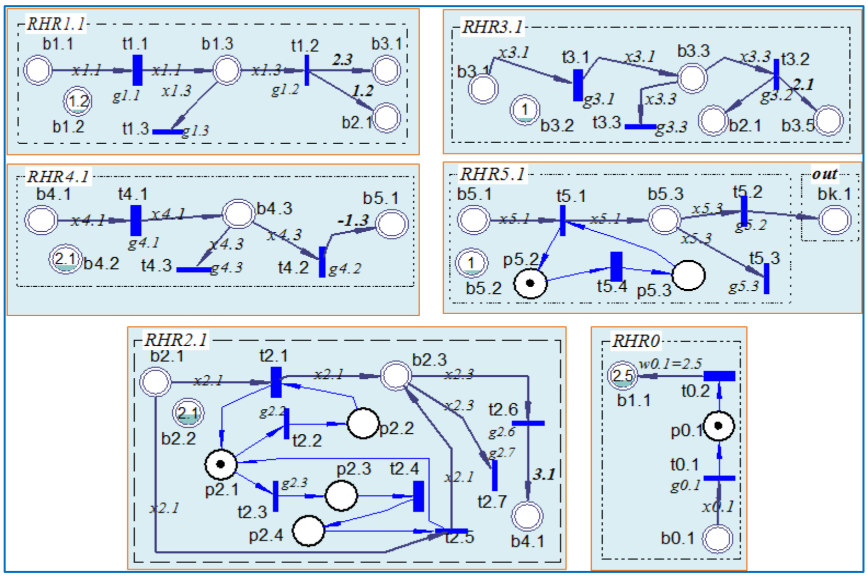


Figure 11. The template subnets involved in reconfiguration of $RH\Gamma1$. model upon firing of rewriting rules $\rho_{i,1} : \{\rho_{i,1}\} \triangleright RHRi.1$.

In this context, a numerical example may be examined to demonstrate the applicability and utility of the RTHPN proposed approach for simulation and analysis of ESNMC models. With this purpose we performed the visual simulation, using upgraded VPNP Tool [24], of the $H\Gamma2$ flat model with the following values of the timed transitions: $\tau'_{1,1} = 1$, $\tau'_{2,1} = 1$, $\tau_{2,4} = 2$, $\tau'_{3,1} = 1$, $\tau'_{4,1} = 1$, $\tau'_{5,1} = 2$, $\tau_{0,2} = 7$ time units and with weight $w_{0,1} = 2.5$.

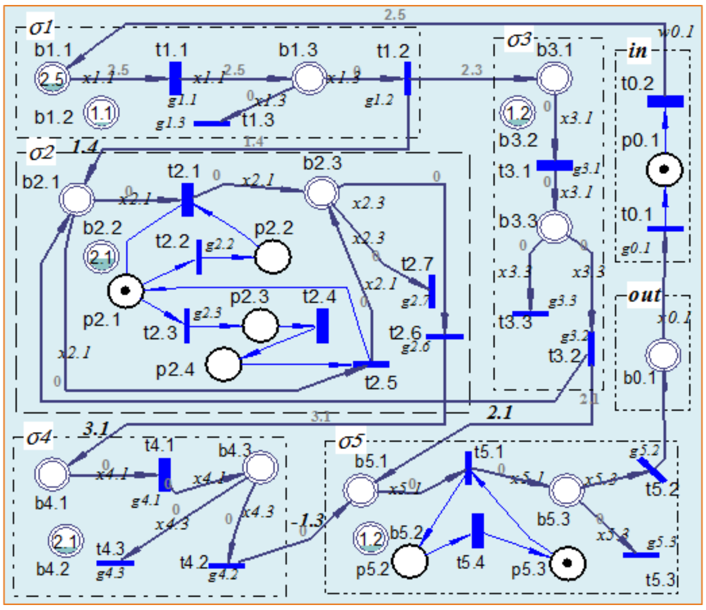


Figure 12. The $H\Gamma 2$ flat model obtained after run-time reconfiguration of the $RH\Gamma 1$ model.

The symbolic reachability graph (RG) of the $H\Gamma 2$ flat model, constructed with these parameters, is shown in the figure 13.

The arc labels of this RG represent step firing sequence set of transitions at time $\tau_k = k, k = 1, 2, \ldots, K$ time units. The meanings of these labels are: $\vartheta_1(\tau_1) = \{t_{1,1}t_{1,2}\}$, $\vartheta_5(\tau_7) = \{t_{0,2}\}$,

for $r_{2,1}$ of $G\Pi 2$: $\vartheta_2^1(\tau_2) = \{t_{2,2}t_{2,1}t_{2,7}\} \| \{t_{3,1}t_{3,2}\} \| \{t_{5,4}\}$, $\vartheta_3^1(\tau_3) = \{t_{2,2}\} \| \{t_{5,1}t_{5,2}\}$, $\vartheta_4^1(\tau_4) = \{t_{2,1}t_{2,7}\}$,

$\vartheta_5^1(\tau_7) = \{t_{0,2}\}$ and for $r_{2,1}$ of $G\Pi 2$: for $r_{2,2}$: $\vartheta_2^2(\tau_2) = \{t_{2,3}\} \| \{t_{3,1}t_{3,2}\} \| \{t_{5,4}t_{5,1}t_{5,2}\} \| \{t_{0,1}\}$,

$\vartheta_3^2(\tau_3) = \{t_{2,4}t_{2,5}t_{2,6}\}$, $\vartheta_4^2(\tau_4) = \{t_{4,1}t_{4,2}t_{5,1}t_{5,3}\}$.

In the label expressions shown above, the operator $\|$ indicates the parallel firing of the respective sequences of transitions. The meanings of discrete places subsets that are marked in RG1 of $H\Gamma 2$ flat model are:

$$\mathbf{m}_1 = \mathbf{m}_4 = (p_{0,1}p_{2,2}p_{5,2}),\ \mathbf{m}_0 = (p_{0,1}p_{2,1}p_{5,2}),\ \mathbf{m}_2 = (p_{0,1}p_{2,1}p_{5,3}),$$

$$\mathbf{m}_3 = (2p_{0,1}p_{2,3}p_{5,2}),\ \ \mathbf{m}_5 = (2p_{0,1}p_{2,1}p_{5,3}),\ \mathbf{m}_6 = (2p_{0,1}p_{2,1}p_{5,2}).$$

The GUI screenshot of VPNP Tool with moment of time that the tokens (spike) occur in the continuous place $b_{0,1}$ and the *history* of the configurations γ_τ with drawing the firing of respective transitions in the $H\Gamma 2$ flat model is shown in Figure 14.

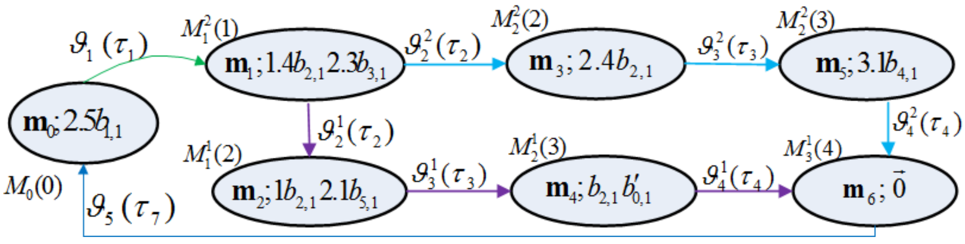
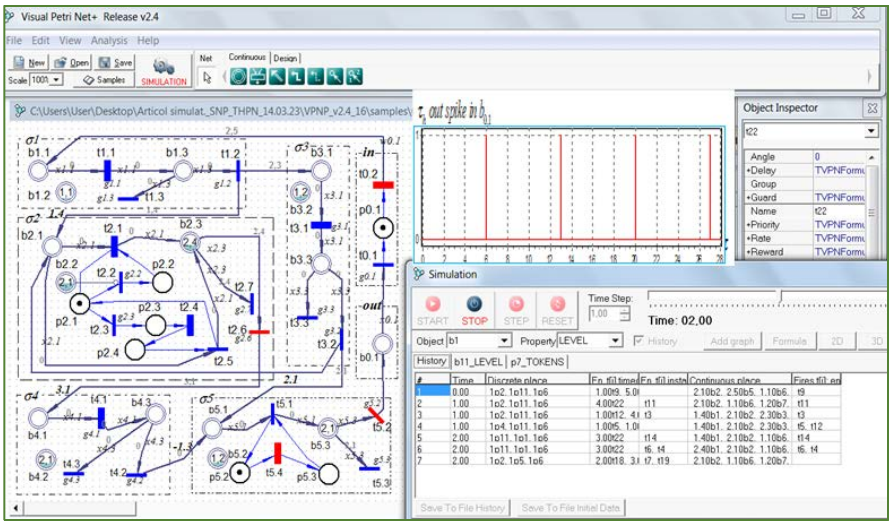


Figure 13. The symbolic reachability graph RG1of the $H\Gamma 2$ flat model.

Thus, the behavior of the $H\Gamma 2$ net is similar to the behavior of the $G\Pi 2$ model and these are the following behaviur properties: *bounded*; *non-safe*; *live*; *deadlock–free*. Immediate transitions $t_{i,2}$ and $t_{i,3}$ (resp. $t_{2,6}$ and $t_{2,7}$) in template subnets $RHRi.1$ (resp. $RHR2.1$) are replaced by immediate rewriting rules $\rho_{i,2}$ and $\rho_{i,3}$ (resp. $\rho_{2,6}$ and $\rho_{2,7}$) with $g_{i,2}^R = g_{i,3}^R = g_{2,6}^R = g_{2,7}^R :=$ "*False*", $i = 1, 3, 4, 5$. We get behavior of $RH\Gamma 1$ model which is equivalent to that of the flat $H\Gamma 2$ model.



#	Time	Discrete place	En. t[i] timed	En. t[i] insta	Continuous place	Fires t[i] en
1	0.00	1p2. 1p11. 1p6	1.00t9. 5.0		2.10b2. 2.50b5. 1.10b6.	t9
2	1.00	1p2. 1p11. 1p6	4.00t22	t11	2.10b2. 1.10b6. 1.20b7.	t11
3	1.00	1p2. 1p11. 1p6	1.00t12. 4.	t3	1.40b1. 2.10b2. 2.30b3.	t3
4	1.00	1p4. 1p11. 1p6	1.00t5. 1.0		1.40b1. 2.10b2. 2.30b3.	t5. t12
5	2.00	1p11. 1p1. 1p6	3.00t22	t14	1.40b1. 2.10b2. 1.10b6.	t14
6	2.00	1p11. 1p1. 1p6	3.00t22	t6. t4	2.40b1. 2.10b2. 1.10b6.	t6. t4
7	2.00	1p2. 1p5. 1p6	2.00t18. 3.	t7. t19	2.10b2. 1.10b6. 1.20b7.	

Figure 14. The GUI screenshot of VPNP Tool with the time tokens (spike) occurences in the continuous place $b_{0,1}$ and the *history* of the configurations γ_τ of $H\Gamma 2$ model.

In order to obtain a more compact $RH\Gamma$ model during the run-time firing of these immediate rewriting rules in flat $H\Gamma 2$ model it is necessary to put $g^R_{i,2} = g^R_{i,3} = g^R_{2,6} = g^R_{2,7} := "True"$, $i = 1, 3, 4, 5$ and to specify these rules to reconfigure the $H\Gamma 2$ model in an abstract form similar to the $RH\Gamma 1$ model. For this purpose we specify the following template subnets when firing these immediate rewriting rules of flat $H\Gamma 2$ model:

$$\rho_{i,2} : RHLi.2 \triangleright RHRi.2,\ \rho_{i,3} : RHLi.3 \triangleright RHRi.3,\ i = 1, 3, 4, 5;$$

$$\rho_{2,6} : RHL2.6 \triangleright RHR2.6,\ \rho_{2,7} : RHL2.7 \triangleright RHR2.7.$$

Due to the volume restrictions of this paper not to present in graphic form the above-mentioned template subnets, they will be rendered in analytical form, using descriptive expressions (DE), the syntax and semantics of which are described in the papers [16, 26, 29].

Hence, the respective descriptive expressions (DEs) of these left-hand side and right-hand side template subnets of flat $H\Gamma 2$ model are:

$$DE_{RHLi.2} = DE_{RHLi.3} = \{t_{i,1}, b_{i,2}, b_{i,3}, \rho_{i,2}, \rho_{i,3}\}, i = 1, 3, 4;$$

$$DE_{RHL5.2} = DE_{RHL5.3} = \{t_{5,1}, t_{5,4}, p_{5,2}, p_{5,3}, b_{5,2}, b_{5,3}, \rho_{5,2}, \rho_{5,3}\};$$

$$DE_{RHL2.6} = DE_{RHL2.7} = \{t_{2,1}, t_{2,2}, t_{2,3}, t_{2,4}, p_{2,1}, p_{2,2}, p_{2,3}, p_{2,4}, b_{2,2}, b_{2,3}, \rho_{2,6}, \rho_{2,7}\};$$

$$DE_{RHLi.2} = DE_{RHLi.3} = \{t_{i,1}, b_{i,2}, b_{i,3}, \rho_{i,2}, \rho_{i,3}\}, i = 1, 3, 4;$$

$$DE_{RHR1.2} = \tilde{b}_{1,1}[x1.1]_{ge_{1.1}}|_{\rho_{1,1}} (2.3b_{3,1}[2.3]\lozenge 1.4b_{2,1}[1.4]),\ DE_{RHR\,2.2} = \tilde{b}_{2,1}[x2.1]_{ge_{2.1}}|_{\rho_{2,1}} 3.1b_{4,1}[3.1],$$

$$DE_{RHR3.2} = \tilde{b}_{3,1}[x3.1]_{ge_{3.1}}|_{\rho_{3,1}} (1.0b_{2,1} \lozenge 2.1b_{5,1}[2.1]),\ DE_{RHR\,4.2} = \tilde{b}_{4,1}[x4.1]_{ge_{4.1}}|_{\rho_{4,1}} -1.3b_{5,1}[-1.3];$$

$$DE_{RHR1.3} = \tilde{b}_{1,1}[x1.1]_{ge_{1.1}}|_{\rho_{1,1}} (b_{3,1}[2.3] \lozenge b_{2,1}[1.4]),\ DE_{RHR\,2.3} = \tilde{b}_{2,1}[x2.1]_{ge_{2.1}}|_{\rho_{2,1}} b_{4,1}[3.1],$$

$$DE_{RHR3.3} = \tilde{b}_{3,1}[x3.1]_{ge_{3.1}}|_{\rho_{3,1}} (b_{2,1} \lozenge b_{5,1}[2.1]),\ DE_{RHR\,4.3} = \tilde{b}_{4,1}[x4.1]_{ge_{4.1}}|_{\rho_{4,1}} b_{5,1}[-1.3],$$

$$DE_{RHR\,5.2} = \tilde{b}_{5,1}[x5.1]_{ge_{5.1}}|_{\rho_{4,1}} 1b_{0,1};\ DE_{RHR\,5.3} = \tilde{b}_{5,1}[x5.1]_{ge_{5.1}}|_{\rho_{4,1}} b_{0,1}.$$

We note that one of the most important benefits we obtain from the use of $RH\Gamma$ nets when maping and verifying the discrete-continuous processes of SNMC and ESNMC models is that the structure of these models is similar, very concise and flexible to reconfiguring and changing the quantitative parameters during run-time of these $RH\Gamma$ nets.

A generalization of this approach consists in assuming that the firing times $\tau : E_d \times Bag(p) \to IR^+$ of the respective discrete events $e_j \in E_d$ are random variables whose probability distribution functions $F(\cdot)$ have contained support in the set of non-negative real numbers. In this case we obtain stochastic RTHPNs that allow simulation and performance analysis of the stochastic ESNMC models.

4. Conclusions

For the purpose of efficient formalization, implementation and formal correctness analysis of SNMC and ESNMC models, in this paper we define and describe a new extension of THPN, called rewriting THPN with anti-tokens (in short, RTHPN) having guards for transitions, rewriting rules and implicitly annihilation rule of tokens and anti-tokens in same places. The features of RTHPN accept as well the negative (positive) values for: place capacities; markings of discrete and continuous places; marking-dependent arc cardinalities. The RTHPN model allows its structure and/or attributes to change at run-time depending on its current state and/or the occurrence of some events which permits a hierarchical design. Therefore, functionalities and features of system models can be added gradually in run-time.

In order to perform the visual simulation and check the behavioral properties of the $H\Gamma$ flat models, we have developed and integrated into VPNP Tool a special software module. Also, we present a methodology that maps the large SNMC and ESNMC models into compact $RH\Gamma$ and flat $H\Gamma$ nets, which allows analysis of such models via the upgraded VPNP Tool in an easy-to-use manner. The use of RTHPNs in simulation and analysis of ESNMC models is illustrated through an example proving that such approach preserves faithfully their behaviours.

In the future, we will develop and integrate into the VPNP tool special software modules that will allow analysis of RTHPN models involving the simulation of stochastic ESNMC models, Spiking Neural P Systems with Self-Organization and Spiking Neural dP Systems.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ionescu, M.; Păun, Gh.; Yokomori, T. Spiking Neural P Systems. *Fundamenta Informaticae*, 2006, 71 (2, 3), pp. 279-308.
2. Păun, Gh. Computing with Membranes. *Journal of Computer and System Sciences* 2000, 61, pp. 108-143.
3. P systems resource website. Available online: http://ppage.psystems.eu/ (accessed on 12.01.2023).
4. Leporati, A.; Mauri, G.; Zandron, C. Spiking neural P systems: main ideas and results. *Natural Computing* 2022, 21, pp. 629–649.
5. Verlan, S.; Freund, R.; Alhazov, A.; Ivanov, S.; Pan, L. A formal framework for spiking neural P systems. *Journal of Membrane Computing* 2020, 2(4), pp. 355 – 368.
6. Fan, S.; Paul, P.; Wu, T.; Rong H.; Zhang. G. On Applications of Spiking Neural P Systems. *Appled. Sciences* 2020, 10, 7011, pp. 1-26. Doi: 10.3390/app10207011.
7. Rong, H.; Wu, T.; Pan, L.; Zhang, G. Spiking neural P systems: Theoretical results and applications. *Enjoying Natural Computing* 2018, pp. 256–268.
8. Wang, X.; Song, T.; Gong, F.; Zheng, P. On the Computational Power of Spiking Neural P Systems with Self-Organization. June 2016, *Scientific Reports* 2016, 6(1), 27624. Available online: https://www.researchgate.net/publication/303908523 (accessed on 11.01.2023).
9. Ionescu, M.; Păun, Gh.; Perez-Jimenez, M. J.; Yokomori**,** T. Spiking Neural dP Systems, *Fundamenta Informaticae* 2011, 111(4), pp. 423–436.
10. Liu, X.; Ren, Q. Spiking Neural Membrane Computing Models. *Processes* 2021, 9, 733.
11. Lefticaru, R.; Gheorghe, M.; Konur, S.; Niculescu, I.M.; Adorna, H.N. Spiking Neural P Systems Simulation and Verification. In: *Proceedings of the 18-th International Conference on HPCS* 2020, 22-27 Mar 2021, Barcelona, Spain. Available online: http://hdl.handle.net/10454/18683(accessed on 12.01.2023).
12. Dupaya, A. G. S.; Galano, A. C. A. P. ; Cabarle, F.; Cruz, R. T. A. A web-based visual simulator for spiking neural P systems. *Journal of Membrane Computing* 2022, 4, pp. 21–40.

13. Fernandez, A. D. C.; Fresco, R. M.; Cabarle, F. G. C.; Cruz, R. T. A.; Macababayao, I. C. H.; Ballesteros, K, J.; Adorna, H. N. Snapse: A Visual Tool for Spiking Neural P Systems. *Processes* 2021, 9, pp. 1-24.
14. Carandang, J. P.; Villaflores, J.M. ; Cabarle, F. G. C.; Adorna, H. N.; Martinez-del- Amor, M. A. CuSNP: Spiking neural P systems simulators in CUDA. *Romanian Journal of Information Science and Technology* 2017, 20(1), pp. 57–70.
15. Kleijn, J.; Koutny, M. Petri nets and Membrane computing. In: *The Oxford Handbook of Membrane Computing*, Gh. Păun, G. Rozenberg, A. Salomaa, eds., Oxford University Press, 2010, pp. 389-412.
16. Guţuleac E. Descriptive Timed Membrane Petri Nets for Modelling of Parallel Computing. *International Journal of Computers, Communications & Control* 2006, 3 (1), pp. 33-39**.**
17. Metta, V. P.; Krithivasan, K.; Garg, G. Modeling spiking neural P systems using timed Petri nets. In: *NaBIC IEEE Conference* 2009, pp. 25-30.
18. Metta, V. P.; Krithivasan, K.; Garg, D. Modelling and Analysis of Spiking Neural P Systems with Anti-spikes using PNetLab. *Nano Communication Networks* 2011, 2(2-3), pp. 141-149.
19. Petri Nets Tools Database Quick Overview. Available online: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/quick.html (accessed on 12.01.2023).
20. Alla, A.; David, H. Continuous and hybrid Petri nets. *Journal of systems Circuits and Computers* 1998, 8(1), pp. 159-188.
21. Horton, G., Kulkarni, V. G., Nicol, D. M., Trivedi, K. S. Fluid stochastic Petri nets: Theory, application, and solution techniques. *Eur. J. Op. Res.* 1998, 105(1), pp. 184–201.
22. Chiola, G.; Ajmone- Marsan, M.; Balbo, G.; Conte, G. Generalized stochastic Petri nets: A definition at the net level and its implications. *IEEE Transactions on Software Engineering* 1993, 19 (2), pp. 89-107.
23. Guţuleac E. Dynamic rewriting generalized differential Petri nets for discrete-continuous modeling of computer systems. *Meridian ingineresc* 2006, 3, pp. 27-32.
24. Guţuleac, E.; Boşneaga, C.; Reilean, A. VPNP-Software tool for modeling and performance evaluation using generalized stochastic Petri nets. In: *Proceedings of 6-th International Conference on D&AS-2002*, Suceava, România, 2002, pp. 243-248.
25. Guţuleac, E.; Ţurcanu, Iu.; Palii, D. Performance modeling of computing processes using reconfigurable generalized differential stochastic Petri nets. *Meridian ingineresc* 2013, 4, pp. 23-30.
26. Guţuleac, E. Descriptive compositional HSPN modeling of computer systems. *Annals of the Craiova University, Series: Atomation, Computers, Electronics and Mechatronics, România* 2006, 3 (30), pp. 82-87.
27. Klaus Reinhardt. Reachability in Petri nets with inhibitor arcs. *Electronic Notes in Theoretical Computer Science*, 2008, 223, pp. 239–264.
28. Verbeek, H.M.W.; Wynn, M.T.; Van der Aalst, W.M.P.; Hofstede, A.H.M. Reduction rules for reset/inhibitor nets. *Journal of Computer and System Sciences* 2010, 76, pp. 125–143.
29. Moraru, V.; Guţuleac, E.; Zaporojan, S. Uncertainty modelling of dynamically reconfigurable systems based on rewriting stochastic reward nets with z-fuzzy parameters. *Computer Science Journal of Moldova* 2021, 3(87), pp. 388-406.

Citation: Moraru, V.; Sclifos, A.; Sclifos, E.; Guțuleac, E. Simulation and analysis of spiking neural membrane computing models based on rewriting timed hybrid Petri nets with anti-tokens. *Journal of Engineering Science* 2023, 30 (3), pp. 78-98. https://doi.org/10.52326/jes.utm.2023.30(3).06.

Submission of manuscripts: jes@meridian.utm.md