PNlib – A Modelica Library for Simulation of Biological Systems Based on Extended Hybrid Petri Nets

Sabrina Proß[1], Sebastian Jan Janowski[2], Bernhard Bachmann[1],
Christian Kaltschmidt[3], and Barbara Kaltschmidt[3]

[1]University of Applied Sciences Bielefeld, Department of Engineering and Mathematics, Bielefeld, Germany
{sabrina.pross, bernhard.bachmann}@fh-bielfeld.de
[2]Bielefeld University, Faculty of Technology, Bielefeld, Germany
{sebastian.janowski}@uni-bielefeld.de
[3]Bielefeld University, Faculty of Biology, Bielefeld, Germany
{c.kaltschmidt,barbara.kaltschmidt}@uni-bielefeld.de

Abstract. We present a new Petri net simulation environment to enable the processing of experimental data to gain usable new insights about biological systems. Therefore, a powerful mathematical modeling concept – xHPNbio (extended Hybrid Petri Nets for biological applications) – has been defined which is properly adapted to the demands of biological processes. This specification is used for the PNlib (Petri Net library), realized by means of the object-oriented modeling language Modelica, which can be easily integrated for simulation processing in any other network modeling tool, as described in the last part of this paper. There, we briefly describe VANESA, a user-friendly biological network-modeling tool that uses the PNlib and the xHPNbio formalism for the simulation of biological networks.

Keywords: hybrid systems, Petri nets, biological processes, Modelica, xHPN, xHPNbio, VANESA, PNlib.

1 Introduction

Modern computer techniques and large memory capacities make it possible to produce an enormous amount of molecular data stored in huge databases. This data is indispensable for the scientific progress but does not necessarily lead to insight about the functionality of biological systems. To improve the understanding of molecular mechanisms, modern techniques focus on network analysis. The question which is posed here is, what model formalism is appropriate and which simulator? Numerous model formalisms have been proposed for modeling and simulation biological systems (see e.g. [1]). Generally, a distinction must be made between qualitative and quantitative approaches. Qualitative models represent only the fundamental compounds, their interaction mechanisms, and the relationships between them while quantitative models describe, in addition, the time-related changes of the components.

Furthermore, quantitative model formalisms can be divided into discrete and continuous approaches as well as deterministic and stochastic techniques.

In the recent years, Petri nets with their various extensions are becoming increasingly popular. They have been proven to be as universal graphical modeling concept for representing biological systems in nearly all degrees of abstraction. They support the qualitative modeling approach as well as the quantitative one. Furthermore, the biological processes can be modeled discretely as well as continuously and, in addition, discrete and continuous processes can also be combined within a Petri net model to so-called hybrid Petri nets first introduced by David and Alla (e.g. [2]). The Petri net formalism with all its extensions is so powerful that all other formalisms are included. Hence, only one formalism is needed regardless of the approach (qualitative vs. quantitative, discrete vs. continuous vs. hybrid, deterministic vs. stochastic) which is appropriate for the respective system. The Petri net formalism is easy to understand for researchers from different disciplines. It is such an ideal way for intuitive representing and communicating experimental data and new knowledge of molecular mechanisms. Besides, Petri nets allow hierarchical structuring of models and therefore offer the possibility of different detailed views for every observer of the model.

Despite several works and publications with Petri net approaches, there is a serious problem relating to the lacking unity of concepts, notations, and terminologies. Therefore, to show the research community the power of Petri nets, we have analyzed the demands for carefully modeling biological systems and specially developed a Petri net formalism which is called xHPNbio (extended Hybrid Petri Net for biological applications).

This Petri net concept is the specification of the new simulator based on the object-oriented modeling language Modelica. A user-friendly graphical model reconstruction in addition to a well-prepared visualization of simulation results is achieved by connection the new Petri net simulator to VANESA, a powerful and easy-to-use biological modeling tool [3]. This new approach is already in use in the area of dynamic system modeling for hypothesis generation and testing of reconstructed database and lab-validated biological networks.

2 Related Works

Reddy et al. proposed the application of Petri net formalism (introduced by Carl Adam Petri in 1962) for biological network modeling in order to represent and analyze metabolic pathways in a qualitative manner [4]. Thereby, places represent biological compounds such as metabolites, enzymes, and cofactors which are part of biochemical reactions. These biochemical reactions are modeled by transitions and their stoichiometry is represented by the arc weights. Besides, the tokens indicate the presence of compounds.

Moreover, Hofestädt and Thelen expanded the approach of Reddy by introducing functional Petri nets to enable quantitative modeling of biochemical networks [5]. Thereby, the arc weights are functions, which depend on concrete markings of places in order to model kinetic effects.

Due to the fact that a random behavior of molecular reactions at low concentrations has been observed in many experiments, Goss and Peccoud introduced stochastic Petri nets [6]. A stochastic transition does not fire instantaneously but rather with a time delay following an exponential distribution which may depend on the token numbers of the places.

A reasonable way for modeling concentrations of biological compounds is by places containing real token numbers instead of integers and transitions which fire as a continuous flow specified by an assigned speed. The transformation from the discrete to the continuous Petri net concept was first introduced by David and Alla in 1987 and they replaced the term token by mark because tokens relate mostly to integer quantities [7].

Furthermore, Alla and David and proposed combing the discrete and the continuous Petri net concept to so-called hybrid Petri nets [8]. A hybrid Petri net contains discrete places with integer tokens and discrete transitions with time delays as well as continuous places with non-negative real marks and continuous transitions with firing speeds .Matsuno et al. used this approach for modeling gene regulatory networks by discrete and continuous processes [9]. They improved this approach further by adding the properties of functional Petri nets to it so that the arcs as well as the speeds of the transitions are functions depending on the marks of the places [10]. In addition, they extended the hybrid functional Petri nets by two specific arcs, called test and inhibitor arcs [10], to accomplish the modeling of inhibition and activation mechanisms of biological reactions. Chen and Hofestädt as well as Doi et al. demonstrated the applicability of this approach by modeling molecular networks [11, 12]. Moreover, Nagasaki et al. extended the hybrid functional Petri nets further by types with which various data types can be regarded in order to model more complex biological processes which involve various kinds of biological information and data [13]. They called this approach hybrid functional Petri nets with extensions (HFPNe).

Despite these mentioned works and publications, there is a serious problem regarding the lacking unity of concepts, notations, and terminologies. The definition of Petri nets is not standardized; every author has his/her own definitions which are partly not precise enough, not common, or contradictory. Hence, to show the research community the power of Petri nets, they have to be defined precisely together with the corresponding processes, which are essential for the simulation. This has been done in this paper; based on the mentioned Petri net concepts, formalism has been developed which is able to represent nearly all kinds of biological processes. It is called xHPNbio (extended Hybrid Petri Nets for biological applications).

Two common tools are already available for modeling biological processes with the Petri net formalism. The first one is the commercial tool Cell Illustrator and the second one is the freely available tool Snoopy.

The Cell Illustrator is a commercial, widely-used tool available as a Java Web Start application that enables to draw, model, elucidate, and simulate complex biological processes and systems based on extended hybrid functional Petri nets [14]. Discrete and continuous processes can be connected to perform hybrid simulations. The drawback of the Cell Illustrator is that the simulation is like a “black box”. There is no information about how the Petri nets and the corresponding processes are defined

which are necessary for modeling and simulation, e.g. how conflicts in Petri nets are resolved, how the hybrid simulation is performed, and which integrators are used. In addition, there is no possibility to adapt solver settings in order to achieve reliable simulation results.

Snoopy is a freely available unifying Petri net framework to investigate biomolecular networks [15]. A Petri net can be modeled time-free (qualitative model) or its behavior can be associated with time (quantitative model) such as stochastic, continuous, and hybrid Petri nets; thereby, different models are convertible into each other. It is also possible to structure the models hierarchically in order to manage complex networks. The drawback of Snoopy is that a continuous Petri net is interpreted as a graphical representation of a system of ordinary differential equations. Hence, the general Petri net property of non-negative marks cannot be held during simulation. Additionally, conflict situations of hybrid Petri nets are trapped not completely and, thus, negative markings can occur. Furthermore, places cannot be provided with capacities and no functions can be assigned to arcs in hybrid Petri nets.

Hence, these problems led to the development of a new Petri net simulation environment specified by the established xHPNbio formalism. The xHPNbio elements are modeled object-oriented which allows an easy way to maintain, extend, and modify them. Furthermore, the hybrid simulation is performed by an appropriate Modelica-tool. With this several solver settings can be adapted in order to achieve reliable simulation results. Moreover, the xHPNbio formalism is already integrated in VANESA, an easy-to-use biological modeling tool. Using VANESA scientists are able to reconstruct and simulate biological pathways either by drag-and-drop or by loading networks from databases and transforming in the appropriate xHPNbio formalism in one software application.

3 Extended Hybrid Petri Nets for Biological Applications (xHPNbio)

The xHPNbio formalism comprises three different processes, called transitions: discrete, stochastic, and continuous, two different states, called places: discrete and continuous, and four different arcs: normal, inhibition, test, and read arc.

Discrete places contain a non-negative integer quantity, called tokens or *marks* while continuous places contain a non-negative real quantity, called marks. These marks initiate transitions to fire according to specific conditions. These firings lead mostly to changes of the marks in the connected places.

Discrete transitions are provided with *delays* and *firing conditions* and fire first when the associated delay is passed and the conditions are fulfilled. These fixed delays can be replaced by exponentially distributed random values, then, the corresponding transition is called *stochastic transition*. Thereby, the characteristic parameter λ of the exponential distribution can depend functionally on the markings of several places (cp. [16]) and is recalculated at each point in time when the respective

transition becomes active or when one or more markings of involved places change[1]. Based on the characteristic parameter, the next putative firing time $\tau = time + \mathrm{Exp}(\lambda)$ of the transition can be evaluated and it fires when this point in time is reached.

Both – discrete and stochastic transitions - fire by removing the arc weight from all input places and adding the arc weight to all output places. On the contrary, the firing of continuous transitions takes places as a continuous flow determined by the firing speed which can depend functionally on markings and/or time. Places and transitions are connected by "normal" arcs which are weighted by non-negative integer and real numbers, respectively. But also functions can be written at the arcs depending on the current markings of the places and/or time.

Places can also be connected to transitions by test, inhibition, and read arcs. Then their markings do not change during the firing process. In the case of test and inhibitor arcs, the markings are only read to influence the time of firing while read arcs only indicate the usage of the marking in the transition, e.g. for firing conditions or speed functions. If a place is connected to a transition by a test arc, the marking of the place must be greater than the arc weight to enable firing. If a place is connected to a transition by an inhibitor arc, the marking of the place must be less than the arc weight to enable firing. In both cases the markings of the places are not changed by firing. The same place can be connected to the same transition by a test and, in addition, by a normal arc as well as by an inhibitor and normal arc. These arcs are called *double arcs*.

It is important to mention that a discrete transition always fires in a discrete manner by removing and adding marks after a delay is passed regardless of whether a discrete or a continuous place is connected to it. However, a continuous transition always fires in a continuous flow so that a discrete place can only be connected to continuous transitions if it is input as well as output of the transition with arcs of the same weight. In this way, the continuous transition can only be influenced by the discrete place but the discrete marking cannot be changed by continuous firing. Hence, the conversion from discrete to continuous markings and vice versa is always performed by discrete transitions connected to continuous places.

A formal definition of an xHPNbio is given below. Therefore, at first the xHPN-formalism is introduced which is then expanded to xHPNbio by providing the Petri net elements with a biological meaning.

Definition 1. The tuple $(PD, PC, TD, TS, TC, F, G, \mathcal{T}, \mathcal{I}, \mathcal{R}, f, c_l, c_u, e, \wp, d, h, v, s, m_0)$ is a *xHPN* if

— $PD = \{pd_1, pd_2, \dots, pd_{pd}\}$ is a finite set of discrete places,
— $PC = \{pc_1, pc_2, \dots, pc_{pc}\}$ is a finite set of continuous places,
— $TD = \{td_1, td_2, \dots, td_{td}\}$ is a finite set of discrete transitions,
— $TS = \{ts_1, ts_2, \dots, ts_{ts}\}$ is a finite set of stochastic transitions,

[1] The involved places may change their markings only in a discrete manner. Continuous changes of involved places are not allowed because then the putative firing times have to be recalculated the whole time as the continuous change takes place.

- $TC = \{tc_1, tc_2, \dots, tc_{tc}\}$ is a finite set of continuous transitions,
- $PD, PC, TD,$ TS, and TC are pairwise disjoint,
- $F \subseteq (PD \times TD \cup PD \times TS \cup PD \times TC \cup PC \times TC \cup PC \times TD \cup PC \times TS)$ is a set of normal arcs from places to transitions, where $(p_i \to t_j)$ denotes the arc from place p_i to transition t_j,
- $G \subseteq (TD \times PD \cup TD \times PC \cup TS \times PD \cup TS \times PC \cup TC \times PC \cup TC \times PD)$ is set of normal arcs from transitions to places, where $(t_j \to p_i)$ denotes the arc from transition t_j to place p_i,
- $\mathcal{T} \subseteq (PD \times TD \cup PD \times TS \cup PD \times TC \cup PC \times TC \cup PC \times TD \cup PC \times TS)$ is a set of test arcs, where $(p_i \to t_j)_{\mathcal{T}}$ denotes the test arc from p_i to t_j,
- $\mathcal{I} \subseteq (PD \times TD \cup PD \times TS \cup PD \times TC \cup PC \times TC \cup PC \times TD \cup PC \times TS)$ is a set of inhibitor arcs, where $(p_i \to t_j)_{\mathcal{I}}$ denotes the inhibitor arc from p_i to t_j,
- $\mathcal{R} \subseteq (PD \times TD \cup PD \times TS \cup PD \times TC \cup PC \times TC \cup PC \times TD \cup PC \times TS)$ is a set of read arcs, where $(p_i \to t_j)_{\mathcal{R}}$ denotes the read arc from p_i to t_j,
- $f: (F \cup G \cup \mathcal{T} \cup \mathcal{I}, \mathrm{m}) \to \{\mathbb{N}_0 : p_i \in PD, \mathbb{R}_{\geq 0} : p_i \in PC\}$ is an arc weight function which assigns every arc connected to a discrete place a non-negative integer and all others a non-negative real number depending on a concrete marking m, where $f(p_i \to t_j)$ denotes the weight of the arc from place p_i to transition t_j,
- if $p_i \in PD$, $t_j \in TC$ then $(p_i \to t_j) \in F$ if and only if $(t_j \to p_i) \in G$ and $f(p_i \to t_j) = f(t_j \to p_i)$,
- $c_l: \{PD \to \mathbb{N}_0, PC \to \mathbb{R}_{\geq 0}\}$ are the minimum capacities of the places,
- $c_u: \{PD \to \mathbb{N}_0, PC \to \mathbb{R}_{\geq 0}\}$ are the maximum capacities of the places,
- $e: (PD \cup PC) \to \{prio, prob\}$ are the resolution types of the places for type-1-conflicts either priority or probability resolution,
- $\mathcal{p}: (F \cup G) \to \{\mathbb{N} : e(p_i) = prio \wedge t_j \in TD,\ (F \cup G) \to [0,1] : e(p_i) = prob \wedge t_j \in TD\}$ is an enabling function which assigns every arc connected to a discrete transition t_j either a priority or a probability according to the resolution type of the place p_i,
- if $e(p_i) = prio$ then $\mathcal{p}(p_i \to t_k) \neq \mathcal{p}(p_i \to t_l)\ \forall t_k, t_l \in TD_{out}(p_i)$ and $\mathcal{p}(t_k \to p_i) \neq \mathcal{p}(t_l \to p_i)\ \forall t_k, t_l \in TD_{in}(p_i)$, if $e(p_i) = prob$ then $\sum_{t_k \in TD_{out}(p_i)} \mathcal{p}(p_i \to t_k) = 1$ and $\sum_{t_k \in TD_{in}(p_i)} \mathcal{p}(t_k \to p_i) = 1$,
- $d: TD \to \mathbb{R}_{\geq 0}$ is a delay function which assigns every discrete transition a positive, real-valued delay,
- $h: (TS, m) \to \mathbb{R}_{\geq 0}$ is a hazard function which assigns every stochastic transition a positive, real-valued random delay depending on a concrete marking m,
- $v: (TC, m) \to \mathbb{R}_{\geq 0}$ is a maximum speed function which assigns every continuous transition a positive, real-valued maximum speed depending on a concrete marking m,
- $s: (TD \cup TS \cup TC, mv) \to \{true, false\}$ is a condition function which assigns every transition a condition depending on all possible model variables (mv) e.g. time,
- $m_0: \{PD \to \mathbb{N}_0, PC \to \mathbb{R}_{\geq 0}\}$ is the initial marking which must satisfy the condition $c_l(p_i) \leq m_0(p_i) \leq c_u(p_i)\ \forall p_i \in (PD \cup PC)$.

Definition 2. An *xHPNbio* is an xHPN (see Definition 1) with a concrete transformation of xHPN elements to biological ones. This transformation is summarized in the following table by mentioning also some examples of the biological meaning.

xHPN	Biological meaning
Places	*Biological compounds*
	metabolites, enzymes, substances, substrates, products, signals, genes, proteins, cells, complexes, activators, inhibitors, repressors, RNA
Transitions	*Biological processes*
	biochemical reactions, metabolic reactions, interactions, regulatory reactions, signal transduction reactions, chemical reactions, binding, phosphorylation
Marks	*Quantities of biological compounds*
	molecules, concentrations, cells
Normal Arcs	*Connections of biological compounds and processes*
Test arcs	*Activation of biological processes*
	transcription process, activation in gene regulation, enzyme activity, activation mechanisms
Inhibitor arcs	*Inhibition of biological processes*
	repression of gene regulation, inhibition mechanisms
Read arcs	*Needs for biological processes*
	catalysis
Arc weights	*Biological coefficients*
	stoichiometric coefficients, yield coefficients
Min/max. capacities	*Reasonable biological capacities*
	biological knowledge
Delays	*Duration of biological processes*
Hazard functions	*Random duration of biological processes*
	stochastic kinetics
Maximum speeds	*Rate of biological processes*
	kinetics effects/laws
xHPNbio	*Biological systems*
	metabolic networks, signal transduction networks, regulatory networks, chemical networks, cell cycle, cell communication, diseases, population dynamics, flux networks, cultivation processes

This xHPN formalism has been transformed to the modeling language Modelica (see section 4) to enable graphical modeling, hybrid simulation, and animation. The execution of a hybrid simulation requires the definition for activating and firing transitions as well as the resolution of possible conflicts.

A *discrete/stochastic transition* t_j in an xHPN is *active* if the markings of all input places $\left(P_{in}(t_j)\right)$ do not fall below the minimum capacities when the arc weights are

removed, and the maximum capacities of all output places $\left(P_{out}(t_j)\right)$ may not be exceeded when the arc weights are added. Additionally, the input places connected by test arcs must have more marks than the arc weights and the places connected by inhibitor arcs must have less marks than the arc weights; read arcs do not influence the activation of a transition.

However, the activation process of *continuous transitions* requires a differentiation between connected continuous and discrete places. A continuous transition t_j is active if all continuous input places $\left(PC_{in}(t_j)\right)$ have either a marking greater than their minimum capacities or they are fed by at least one input transition, i.e. the input speed I_i is not zero. Additionally, all continuous output places $\left(PC_{out}(t_j)\right)$ have either a marking less than their maximum capacities or they are emptied by at least one output transition, i.e. the output speed O_i is not zero. The connected discrete places have to fulfill the same conditions as mentioned above for activating a discrete transition. In addition, the markings of input places connected by test arcs have to be greater than the arc weights and markings of places connected by inhibitor arcs have to be less than the arc weights.

Definition 3. The tuple $(PD, PC, TD, TS, TC, F, G, \mathcal{T}, \mathcal{I}, \mathcal{R}, f, c_l, c_u, e, \wp, d, h, v, s, m_0)$ is an xHPN. A *discrete/stochastic transition* $t_j \in (TD \cup TS)$ is *active* if and only if

$$\forall\, p_i \in P_{in}(t_j) : \begin{cases} m(p_i) - f(p_i \to t_j) \geq c_l(p_i) & if\ (p_i \to t_j) \in F \\ m(p_i) > f\left((p_i \to t_j)_{\mathcal{T}}\right) & if\ (p_i \to t_j)_{\mathcal{T}} \in \mathcal{T} \\ m(p_i) < f\left((p_i \to t_j)_{\mathcal{I}}\right) & if\ (p_i \to t_j)_{\mathcal{I}} \in \mathcal{I}, \end{cases}$$

and

$$\forall\, p_i \in P_{out}(t_i) :\ m(p_i) + f(t_j \to p_i) \leq c_u(p_i)$$

and the condition s_j must be fulfilled.
A *continuous transition* $t_j \in TC$ is *active* if and only if

$\forall\, p_i \in PC_{in}(t_j)$:

$$\begin{cases} m(p_i) > c_l(p_i) \vee (m(p_i) = c_l(p_i) \wedge I_i > 0) & if\ (p_i \to t_j) \in F \\ m(p_i) > f\left((p_i \to t_j)_{\mathcal{T}}\right) & if\ (p_i \to t_j)_{\mathcal{T}} \in \mathcal{T} \wedge (p_i \to t_j) \notin F \\ m(p_i) > f\left((p_i \to t_j)_{\mathcal{T}}\right) \vee \left(m(p_i) = f\left((p_i \to t_j)_{\mathcal{T}}\right) \wedge I_i > 0\right) & if\ (p_i \to t_j)_{\mathcal{T}} \in \mathcal{T} \wedge (p_i \to t_j) \in F \\ m(p_i) < f\left((p_i \to t_j)_{\mathcal{I}}\right) & if\ (p_i \to t_j)_{\mathcal{I}} \in \mathcal{I}, \end{cases}$$

and

$$\forall\, p_i \in PC_{out}(t_j) :\ m(p_i) < c_u(p_i) \vee (m(p_i) = c_u(p_i) \wedge O_i > 0).$$

and

$$\forall\, p_i \in PD_{in}(t_j) : \begin{cases} m(p_i) - f(p_i \to t_j) \geq c_l(p_i) & if\ (p_i \to t_j) \in F \\ m(p_i) > f\left((p_i \to t_j)_{\mathcal{T}}\right) & if\ (p_i \to t_j)_{\mathcal{T}} \in \mathcal{T} \\ m(p_i) < f\left((p_i \to t_j)_{\mathcal{J}}\right) & if\ (p_i \to t_j)_{\mathcal{J}} \in \mathcal{J}, \end{cases}$$

and

$$\forall\, p_i \in PD_{out}(t_j) :\ m(p_i) + f(p_i \to t_j) \leq c_u(p_i)$$

and the condition s_j must be fulfilled.

An active transition has to be enabled by all input and output places to become firable. Thereby, enabled discrete transitions wait until the assigned delay is elapsed and stochastic transitions fire first when the putative firing time is reached. However, continuous transitions fire immediately when they are enabled.

Several conflicts can occur when the places have to enable their connected active transitions. Possibly, a discrete place or a continuous place connected to discrete transitions has not enough marks to enable all output transitions simultaneously or cannot receive marks from all active input transitions due to the maximum capacity. Then a conflict arises that has to be resolved (*type-1-conflict*). This can be either done by providing the transitions with priorities or probabilities. In the first case, a deterministic process decides which place enables which transitions and in the second case the enabling is performed at random; thereby transitions assigned with a high probability are chosen preferentially.

Another conflict can occur between a continuous place and two or more continuous transitions when the input speed is not sufficient to fire all output transitions with the instantaneous speed $\tilde{v}_j$ (see equation (1)) (*type-2-output-conflict*) or when the output speed is not sufficient to fire all input transitions with the speed of equation (1) (*type-2-input-conflict*). This conflict is solved by sharing the speeds proportional to the assigned maximum speeds (see [17]).

If a conflict occurs between a place and continuous as well as discrete/stochastic transitions, the discrete/stochastic transitions take always priority over the continuous transitions (*type-3-conflict*).

A last conflict can occur when a discrete place has not enough marks to enable all connected continuous transitions (*type-4-conflict*). This is solved by prioritization of the involved transitions.

The *firing* is then performed in the following way. Discrete transitions fire by removing as much marks as the arc weights from all input places and by adding as many marks as the arc weights to all output places. However, the firing process of continuous transitions take place as a continuous flow with a maximum speed assigned to every transition. The recalculation of a *discrete marking* is described by an algebraic equation while a *continuous marking* is recalculated by a differential equation describing the flow of the continuous firing and an algebraic equation representing the firings of discrete transitions.

Definition 4. The tuple $(PD, PC, TD, TS, TC, F, G, \mathcal{T}, \mathcal{I}, \mathcal{R}, f, c_l, c_u, e, \wp, d, h, v, s, m_0)$ is an xHPN. The *firing process* of an active *continuous transition* $t_j \in TC$ is described by a negative mark change of all continuous input places which is expressed by the differential equation

$$\frac{dm(p_i)}{dt} = -f(p_i \to t_j) \cdot \tilde{v}_j \qquad \forall\, p_i \in PC_{in}(t_j)$$

and a positive mark change of all continuous output places which is expressed by the differential equation

$$\frac{dm(p_i)}{dt} = f(t_j \to p_i) \cdot \tilde{v}_j \qquad \forall\, p_i \in PC_{out}(t_j),$$

where $\tilde{v}_j$ is the *instantaneous speed* of transition t_j and calculated by the following equation if the transition is not involved in a type-2-conflict [2, 17]

$$\tilde{v}_j = \min\left\{ \min_{p_i \in PI_{in}(t_j)} \left(\frac{1}{f(p_i \to t_j)} \sum_{t_k \in TCF_{in}(p_i)} f(t_k \to p_i) \cdot \tilde{v}_k \right), \min_{p_i \in PI_{out}(t_j)} \left(\frac{1}{f(t_j \to p_i)} \sum_{t_k \in TCF_{out}(p_i)} f(p_i \to t_k) \cdot \tilde{v}_k \right), v_j \right\} \quad (1)$$

where $PI_{in}(t_j)$ is the set of continuous input places of t_j with $m(p_i) = c_l(p_i)$, $PI_{out}(t_j)$ is the set of continuous output places of t_j with $m(p_i) = c_u(p_i)$, $TCF_{in}(p_i) \subseteq TC$ is the set of all continuous firing input transitions, $TCF_{out}(p_i) \subseteq TC$ is the set of all continuous firing output transitions. If the transition is involved in a type-2-conflict, this speed has to be adapted appropriately (see [17]).

An active *discrete transition* $t_j \in TD$ waits d_j time units before it *fires* and a *stochastic transition* $t_j \in TS$ fires when the putative firing time τ_j is reached which is calculated based on the hazard function h. Both fire by removing the arc weight from all input places

$$m'(p_i) = m(p_i) - f(p_i \to t_j) \;\; \forall\, p_i \in P_{in}(t_j)$$

and by adding the arc weight to all output places

$$m'(p_i) = m(p_i) + f(t_j \to p_i) \;\; \forall\, p_i \in P_{out}(t_j).$$

The *marking* of a discrete place $p_i \in PD$ can be recalculated by the following algebraic equation

$$m'(p_i) = m(p_i) + \sum_{t_j \in TDF_{in}(p_i)} f(t_j \to p_i) - \sum_{t_j \in TDF_{out}(p_i)} f(p_i \to t_j),$$

whereby $TDF_{in}(p_i) \subseteq (TD \cup TS)$ is the set of all discrete/stochastic firing input transitions and $TDF_{out}(p_i) \subseteq (TD \cup TS)$ is the set of all discrete/stochastic firing output transitions.

The *continuous mark change* of a continuous place $p_i \in PC$ is performed with the aid of the following differential

$$\frac{dm(p_i)}{dt} = \sum_{t_j \in TCF_{in}(p_i)} f(t_j \to p_i) \cdot \tilde{v}_j - \sum_{t_j \in TCF_{out}(p_i)} f(p_i \to t_j) \cdot \tilde{v}_j$$

and, in addition, by the following algebraic equation for the discrete mark change caused by firing connected discrete transitions

$$m_{dis}(p_i) = \sum_{t_j \in TDF_{in}(p_i)} f(t_j \to p_i) - \sum_{t_j \in TDF_{out}(p_i)} f(p_i \to t_j).$$

At these discrete firing times the continuous marking is reinitialized by

$$m'(p_i) = m(p_i) + m_{dis}(p_i).$$

4 The Petri Net Library in Modelica (PNlib)

The xHPN definition and the corresponding definitions for activation, enabling, and firing mentioned above have been implemented by means of the object-oriented modeling language Modelica to enable graphical hierarchical modeling, hybrid simulation, and animation [18]. Modelica is developed and promoted by the Modelica Association since 1996 for modeling, simulation, and programming primarily of physical and technical systems and processes. Additionally, the Modelica standard library is available from the Modelica Association to model mechanical (1D/3D), electrical (analog, digital, machines), thermal, fluid, control systems, and hierarchical state machines. Furthermore, several libraries have been developed in the last decade for specific applications. An overview can be found on the Modelica homepage (www.modelica.org). The development of the language and libraries is ongoing and driven by several European projects (EUROSYSLIB, MODELISAR, OPENPROD, and MODRIO). Since the year 2000, Modelica is used successfully in the industry which is documented in the proceedings of many Modelica conferences and journals.

The Modelica models are described on the textual level by discrete, algebraic, and differential equations and by schematics on the graphical level. A schematic consists of connected *components* which are defined by other components or on the lowest level by equations in the Modelica syntax. The components have connectors which describe the interaction between them. By drawing a line from one component to another, a connection is established to enable interactions. In this manner a model is constructed. Several components can be structured in libraries, called *packages*, which provides hierarchical modeling. Moreover, the *wrapping technique* enables the representation of sub-models consisting of several connected components by a specific adapted icon in order to simplify the modeling process. Then, the sub-models can be used several times in the same or in different models and, in addition, it offers an easy-to-use-model at the top level with an intuitive and familiar adapted view.

For graphical modeling, simulation, and animation an appropriated environment is needed. Several commercial and open- source tools are available. A full list can be found on the Modelica homepage (www.modelica.org).

Each of the xHPN components - transitions, places, and arcs - is modeled by an own Modelica model which are organized and structured in a Modelica package, called *PNlib* (*Petri Net library*). All components are defined by discrete (event-based), algebraic, and differential equations (cp. [19]).

The main process in the place model is the recalculation of the marking after firing a connected transition. In the case of the discrete place model, this is realized by the discrete equation

```
when fire pre(reStart) then
   t = if fire then pre(t)+firingSumIn-firingSumOut
       else reStartTokens;
end when;
```

whereby `pre(t)` accesses the marking `t` immediately before the transitions fire. To this amount, the arc weight sum of all firing input transitions is added and the arc weight sum of all firing output transitions is subtracted from it. Additionally, the tokens are reset to `reStartTokens` when the user-defined condition `reStart` becomes true; this could be a global condition.

The marking of continuous places can change continuously as well as discretely. This is implemented by the following construct

```
der(t) = conMarkChange;
when discreteFire then
   reinit(t, t+discreteMarkChange);
end when;
when reStart then
   reinit(t,reStartMarks);
end when;
```

whereby the `der-operator` access the derivative of the marking `t` according to time. The continuous mark change is performed by a differential equation while the discrete mark change is performed by the `reinit`-operator within a discrete equation. This operator causes a re-initialization of the continuous marking every time when a connected discrete transition fires. Additionally, the marking is re-initialized by `reStartMarks` when the condition `reStart` becomes true.

The main process of the transition is to check if it can fire. When it is possible, discrete and stochastic transitions wait as long as the (random) delay is passed while the continuous transitions fires continuously with a speed calculated in the transition model. Via connector variables, the places report the transitions their markings and the transitions report when they fire. The conflicts which could arise in xHPN models have to be resolved by the mentioned methods to have successful and reliable simulation (see [17]).

5 Application

In this section we briefly demonstrate the PNlib possibilities in the software application VANESA (www.vanesa.sf.net). VANESA is a powerful and easy-to-use network modeling tool to members of the laboratory, which combines different fields of studies such as life science, database consulting, modeling and visualization for a semi-automatic and lab-validated reconstruction of biological networks. The application helps to model experimental results that can be expanded with database information to perform modern biological network analysis. Based on project experimental data and the integrated databases in VANESA, users are able to explore and reconstruct any biological system, which can be further transformed, simulated, and analyzed in the language of Petri nets.

Therefore, VANESA uses the PNlib and xHPNbio formalism for simulation processing. One feature of VANESA is the possibility to automatically transform any kind of network into the language of an xHPNbio. Thus, users are able to model and simulate dynamic systems within one tool. Petri net simulations are performed in the background, not visible to users. Simulation results are visualized in plots and animated within the active window (see Fig. 1).

6 Conclusions

A powerful Petri net simulation environment has been developed to enable the processing of experimental data to usable new insights about biological systems. The mathematical modeling concept xHPNbio serves as specification for the presented simulation environment. The xHPNbio elements are modeled object-oriented by discrete, algebraic, and differential equations in the Modelica language. This allows an easy way to maintain, extend, and modify them. The hybrid simulation is performed by an appropriate Modelica tool which usually comprises several possibilities to adapt the solver settings in order to achieve reliable simulation results.

The mathematical modeling concept xHPNbio, was specially developed based on the demands of biological processes, and is so powerful but also so universal and generic that it is an ideal all-round-tool for modeling and simulation of nearly all kinds of processes such as business processes, production processes, logistic processes, work flows, traffic flows, data flows, multi-processor systems, communication protocols, and functional principals.

The PNlib will be freely available on the Modelica homepage (www.modelica.org) soon. However, in this paper we have already presented a software application, called VANESA, which uses the PNlib for simulation processes of biological networks.

In addition, a future goal is to provide an open source Petri net simulation tool for up till now the PNlib works only with the commercial Modelica tool Dymola. This demands a further development of the open source Modelica tool OpenModelica to get the PNlib to work with it because some Modelica features are not yet supported. The University of Applied Sciences Bielefeld is already closely involved in the further development of the OpenModelica tool [20].

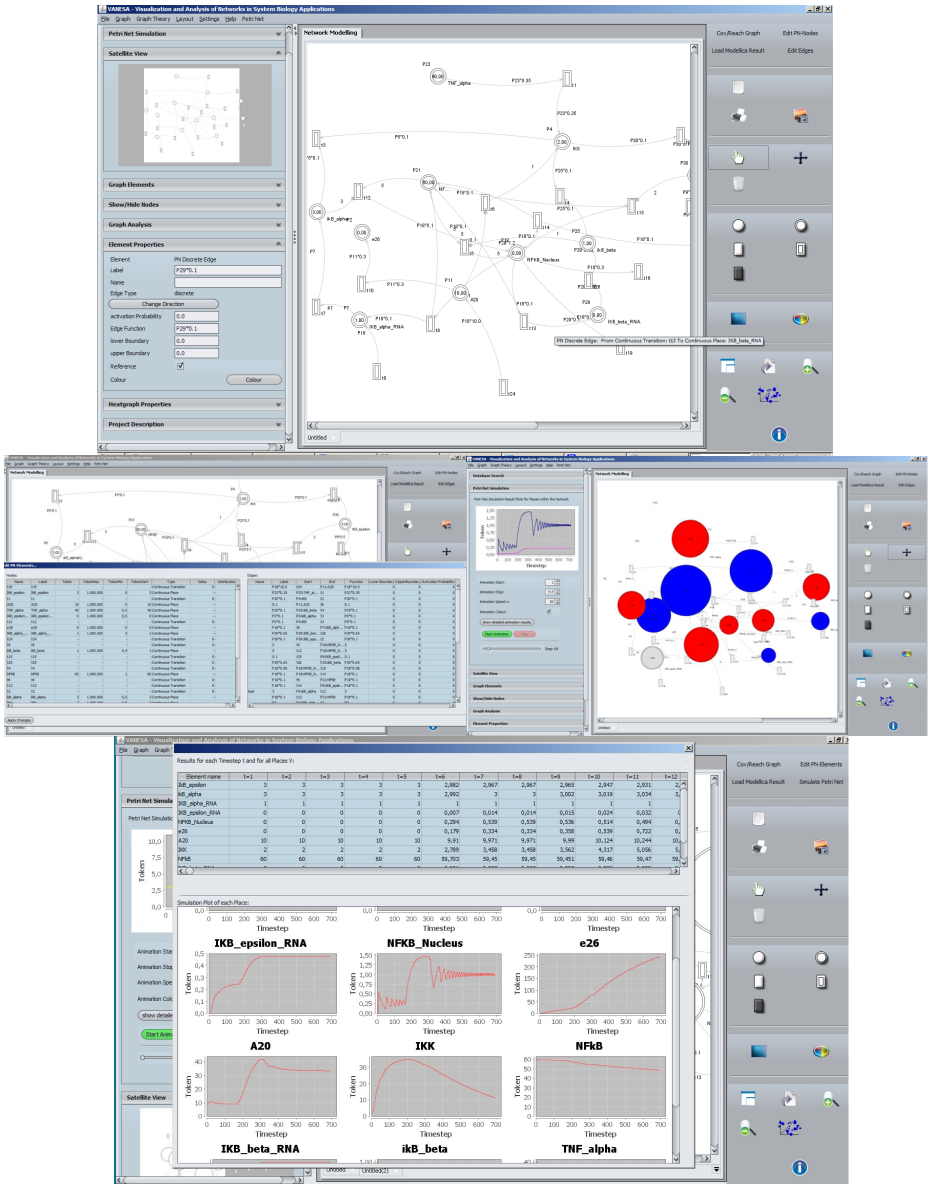

Fig. 1. An example of a simulation in VANESA. Starting point is the reconstruction of a biological network (NF-κB), which is automatically transformed into the xHPNbio formalism. The top picture shows the reconstruction of the biological network by data mining and information fusion. The middle left picture shows the xHPN user settings for the following simulation in the PNlib of Modelica. The middle right picture and the bottom picture show visualized and animated simulation results in VANESA. Simulation results can be plotted and also mapped on each element within the model. Plotted curves give detailed insight into system dynamics, whereas animated simulation results give insights about a certain time step. Shape and color of the elements present amount and developing of values (red increasing, blue decreasing).

References

1. Wiechert W.: Modeling and simulation: tools for metabolic engineering. Journal of biotechnology 94(1), 37–63 (2002)
2. David R., Alla H.: On Hybrid Petri Nets. Discrete Event Dynamic Systems: Theory and Applications(11), 9–40 (2001)
3. Janowski, S.J., Kormeier B., Töpel T., and Hofestädt R.:. Modeling of Cell-to-Cell communication processes with Petri Nets using the example of Quorum Sensing, in: E. Wingender (Ed.), Biological Petri Nets, 162. Amsterdam: IOS Press, 182 - 203 (2011)
4. Reddy V.N., Mavrovouniotis M.L., Liebman M.N.: Petri net representations in metabolic pathways. Proceedings of 1st International Conference on Intelligent Systems for Molecular Biology, 328-336 (1993)
5. Hofestädt R., Thelen S.: Quantitative modeling of biochemical networks. In Silico Biology 1(1),39–53 (1998)
6. Goss P., Peccoud J.: Quantitative modeling of stochastic systems in molecular biology by using stochastic Petri nets. Proceedings of the National Academy of Sciences of the United States of America 95(12), 6750-6755 (1998)
7. David R., Alla H.: Continuous Petri nets. Proceedings of 8th European Workshop on Application and Theory of Petri nets, 275-294 (1987)
8. Alla, H. and David, R., Continuous and hybrid Petri nets, Journal of Circuits, Systems, and Computers, 8:159- 188 (1998)
9. Matsuno H., Doi A., Nagasaki M., Miyano S.: Hybrid Petri net representation of gene regulatory network. Pacific Symposium on Biocomputing 5, 341-352 (2000)
10. Matsuno H., Tanaka Y., Aoshima H., Doi A., Matsui M., Miyano S.: Biopathways representation and simulation on hybrid functional Petri net. In Silico Biology 3(3), 389–404 (2003)
11. Chen M., Hofestädt R.: Quantitative Petri net model of gene regulated metabolic networks in the cell. In Silico Biology 3(3), 347–365 (2003)
12. Doi A., Fujita S., Matsuno H., Nagasaki M., Miyano S.: Constructing biological pathway models with hybrid functional Petri nets. In Silico Biology 4(3), 271–291 (2004)
13. Nagasaki M., Doi A., Matsuno H., Miyano S.: A Versatile Petri Net Based Architecture for Modeling and Simulation of Complex Biological Processes. Genome Informatics 15(1), 180-197 (2004)
14. Nagasaki M., Saito A., Jeong E., Li C., Kojima K., Ikeda E., Miyano S.: Cell Illustrator 4.0: A computational platform for systems biology. In Silico Biology 10(1), 5–26 (2010)
15. Rohr C., Marwan W., Heiner M.: Snoopy—a unifying Petri net framework to investigate biomolecular networks. Bioinformatics 26(7), 974 (2010)
16. Heiner M., Gilbert D., Donaldson R.: Petri nets for systems and synthetic biology. Proceedings 8th International Conference on Formal Methods for Computational Systems Biology, 215–264 (2008)
17. Proß S.: Hybrid Modeling and Optimization of Biological Processes. PhD thesis (in preparation) (2012)
18. Modelica Association: Modelica - A Unified Object-Oriented Language for Physical Systems Modeling Language Specification Version 3.2 (2010)
19. Proß S., Bachmann B.: An Advanced Environment for Hybrid Modeling of Biological Systems Based on Modelica. Journal of Integrative Bioinformatics(8), 1–34 (2011)
20. Braun W., Bachmann B., Proß S.: Synchronous Events in the OpenModelica Compiler with a Petri Net Library Application. Proceedings of 3rd International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, 63–70 (2010)