QATAR UNIVERSITY

COLLEGE OF ENGINEERING

MODELING AND SIMULATION OF BIO-PATHWAYS USING HYBRID FUNCTIONAL PETRI NETS

BY

IMENE NOUREDDINE MECHETER

A Thesis Submitted to the Faculty of

the College of Engineering

in Partial Fulfillment

of the Requirements

for the Degree of

Masters of Science in Computing

January 2017

COMMITTEE PAGE

The members of the Committee approve the Thesis of Imene Mecheter

defended on 10/01/2017.

_____

Prof. Sebti Foufou, Dr. Rachid Hadjidj
Thesis/Dissertation Supervisor

_____

Prof. Salima Hassas
Committee Member

_____

Prof. Ali Jaoua
Committee Member

_____

Dr. Mohammad Saleh
Committee Member

Approved:

_____

Khalifa Al-Khalifa, Dean, College of Engineering

ABSTRACT

MECHETER, IMENE, N., Masters:

January: 2017, Masters of Science in Computing

Title: Modeling and Simulation of Bio-pathways Using Hybrid Functional Petri Nets

Supervisors of Thesis Sebti, Foufou and Rachid, Hadjidj.

The study of biological systems is growing rapidly, and can be considered as an intrinsic task in biological research and a prerequisite for diagnosing diseases and drug development. The integration of biological studies with computer technologies led to a noticeable development in this field with the appearance of many powerful modeling and simulation techniques and tools. The help of computers in biology resulted in deeper knowledge about complex biological systems and biopathways behaviors. Among modeling tools, the Petri Net formalism plays an important role. Petri Net is a powerful computerized and graphical modeling technique originally developed by Carl Adam Petri in 1960 to model discrete event systems. With its various extensions, Petri Nets find applications in many other fields including Biology. The extension known under the name Hybrid Functional Petri Net (HFPN) was developed specifically to model biological systems. Traditionally, biological processes are captured as systems of ordinary differential equations. However, HFPNs offer a much more elegant and versatile approach to represent these processes more accurately. In fact, these nets allow to capture phenomena which are impossible to capture with ordinary differential equations, while being more intuitive to understand and model with. In this work we propose an approach to automatically translate a system of ordinary differential equations representing a biological process into a HFPN. The resulting HFPN not only preserves the semantics of the original model, but is also

more humanly readable thanks to the use of a novel technique to connect its components in a smart way. To validate our approach, we implemented it as an extension to the tool Real Time Studio (an integrated environment for modeling, simulation and automatic verification of real-time systems), and compared our simulation results with those obtained by simulating systems of ordinary differential equations on MATLAB.

DEDICATION

To My Parents,

The reason of what I have became today.

To My Sisters,

My Soulmates that always inspire and support me.

To My Friends,

The companions of the hard journey who continually push me up.

ACKNOWLEDGMENTS

I would like to express my gratitude to my supervisor Dr. Rachid Hadjidj for his guidance, encouragement, sage advice and innumerable support through the learning and working journey of this master thesis. Furthermore, I would like to thank Dr. Sebti Foufou for his insightful comments that were deeply appreciated.

Finally, I would like to thank my parents, my family and my friends who have supported me throughout the entire journey, by keeping me motivated and helping me putting the pieces together.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

1. Introduction

Computer sciences and information technology lead the evolution of different research areas toward a computerized world. Biology is one of the most sophisticated fields that needs the help of computer sciences to solve the complexity of biological systems and bio-pathways. The following sections present a brief about biological systems, bioinformatics, computational biology and computer modeling.

1.1 Biological Systems

Systems Biology is an interdisciplinary field that aims at analyzing and understanding the functioning of integrated biological processes instead of individual biological components [1]. Living organisms interact with each other and build complex biological systems that have various tasks and functionalities. Thousands of complex interactions take place on numerous metabolic and signaling biopathways. Although still there is no clear agreement among researchers on a single precise definition of biopathways, in general, biopathways represent a series of interconnected cellular events, such as signal transduction, enzymatic reaction events, and genetic regulation events [2]. These interconnected events describe the interactions of molecular entities such as genes, proteins and metabolites.

The study of biological systems and their complex pathways has evolved rapidly and has become the primary focus of molecular biology [2]. The complexity of these studies led to the creation of new fields called Bioinformatics and Computational Biology.

1.2 Bioinformatics and Computational Biology

The fields of Bioinformatics and Computational Biology originate from the integration of life sciences with computer sciences. Both fields make life sciences data more understandable with a vision to providing qualitative results and decisions. According to the NIH Biomedical Information Science and Technology Initiative Consortium each field is defined as follows:

Bioinformatics: “Research, development, or application of computational tools and approaches for expanding the use of biological, medical, behavioral or health data, including those to acquire, store, organize, archive, analyze, or visualize such data ” [3].

Computational Biology: “The development and application of data-analytical and theoretical methods, mathematical modeling and computational simulation techniques to the study of biological, behavioral, and social systems” [3].

Nowadays, computer science is leading a vast range of scientific research areas. One main reason behind the development of the Bioinformatics and Computational Biology fields is the huge need for computer systems and software tools in biological research to solve problems related to the complexity of biological systems and pathways. This evolution will efficiently and expeditiously help humans to capture and understand how complex interactions and behaviors of biological processes take place. Therefore, we can say that modeling a biological system is equivalent to developing a computer program.

1.3 Modeling

The first question that may come to one’s mind when reading about modeling is “What do we expect to obtain from a model?” A model is a way to combine system details

into a structure that presents the system in a more understandable way and provides more insights and knowledge about the system's working mechanism. In bioinformatics and computational biology, modeling helps in understanding the mechanisms of diseases, in personalizing drug regimens and developing better drugs. Nowadays, many studies focus on developing and adapting existing modeling and simulation tools to biological systems. Traditionally in biology, models are represented as systems of differential equations in which equations provide deep insights by representing many details and produce quantitative analysis results [4]. However, this modeling approach can be complex for a biologist to compute or analyze in order to understand the model interactions and behaviors.

Different modeling and simulation formalisms have been developed and applied to various types of biological systems. These models help capture the behavior of complicated systems for study, learning or research purposes. For instance, Boolean networks [5,6], stochastic modeling [4], State charts [7], process algebras [8, 9], and Petri Net [10] formalisms are of relevance in thin context and will be presented in Chapter 2.

In this work, we use an extension of the Petri Net formalism called Hybrid Functional Petri Net (HFPN) [11] to model biological pathways by converting a system of ordinary differential equations into a human readable Petri Net representation. The HFPN formalism is considered as one of the most appropriate and representative technique for modeling and simulating biological systems. This formalism offers a graphical and mathematical representation, and is mainly used in computer science to model systems with concurrent properties. More details about Petri Nets and their available extensions and tools are discussed in Chapter 2.

2. Problem Statement

Traditionally, different biopathways models are represented using systems of ordinary differential equations. However, this modeling technique is not easily understood by simple biologists as it requires a good background in mathematics. In addition, differential equations are not expressive enough to fit the modeling requirements of all aspects found in biological systems. In fact, biological pathways are composed of sophisticated interactions that require a more powerful modeling and simulation technique, in addition to using the aid of computer technology.

3. Objective and Significance

The main objective of this work is to develop an approach to convert models written as systems of ordinary differential equations into semantically equivalent Hybrid Functional Petri Net models. The resulting models should be humanly readable, in the sense that resulting Petri net components should clearly map to the different elements and mechanisms involved in the modeled system.

Such an approach can help biologists and researchers minimize the time spent to understand models, identify different diseases mechanisms, enhance research outcomes, and ultimately develop new drugs. These factors would definitely lead to better quality of life and health. Additionally, translated models can be enhanced and extended thanks to the expressiveness of the HFPN formalism.

4. Main Contributions

In this thesis, an automatic conversion approach is developed to translate a system of ordinary differential equations representing a biopathway into a HFPN model. The resulting model is semantically equivalent to the original model and more intuitive to understand thanks to a novel approaches to connect places and transitions. Our approach is general enough, and can be used to convert models to equivalent HFPN models, even if they do not represent biopathways.

To validate our approach, we implemented it as an extension to the tool Real-Time studio [12], were translated models can be simulated for further studies and observations.

5. Document Overview

This document is organized as follows: the literature review and background are presented in chapter 2. Chapter 3 is devoted to the implementation and methodology used in this work. Chapter 4 demonstrates obtained results and how they have been evaluated. The discussion, future directions and conclusion are presented in Chapter 5.

Chapter 2

1. Literature Review

The vast and growing amount of biological data requires combined efforts from the fields of biology, mathematics and computer sciences in order to produce powerful formalisms and representations that help to deal with this rapid development. Due to different properties and complex behaviors of biological systems, a variety of modeling formalisms with different features and extensions have been developed, backed by computer tools to support their use. These formalisms and techniques can be classified into different categories such as verbal models, conceptual or diagrammatic models, physical models and formal models [13]. Each category offers deterministic or stochastic modeling capabilities, as well as qualitative or quantitative analysis capabilities. In the literature, there are several techniques that have been introduced, such as ordinary differential equations, Boolean modeling, states machines, stochastic models, state charts, discrete event modeling (e.g. Petri Nets) and many others that will be discussed in the following sections.

1.1 Modeling and Simulation Formalisms of Biological Systems

Systems of Ordinary Differential Equations

This technique is a traditional representation that has been widely used to represent many biological models. Ordinary Differential Equations (ODE) are used to model real system that evolves in time. Hence, it is a suitable choice for quantitative modeling of many biological processes such as biochemical networks [14].

Stode [15] is a software tool developed for the stochastic simulation of biochemical models represented as systems of ordinary differential equations. The modeling approach is based on constructing a set of differential equations, then solving them to produce a time-course of concentrations of system reactants. The results can be used for predictions and comparisons with experimental results. E-Cell [16] is another tool that provides users with the ability to define biological model functionalities, then simulate the model by numerically integrating differential equations described in the reaction rules, which gives the possibility to observe dynamic changes in the behaviors of cell systems.

Ordinary differential equations are in the core of the well-known System Biology Markup Language (SBML) [17 , 18] which is a standard XML-based format developed to represent various biological models, such as cell signaling pathways, metabolic pathways, gene regulation, and others. This free and open standard representation helps modelers use structure-related methods to formalize any reaction [19].

Despite their widespread use, ordinary differential equations are not powerful enough to represent all aspects of a biological system [5]. In fact purely mathematical models such as ODE are less flexible than computational models which are able to model biological systems at different levels of abstraction [1].

Boolean Networks

Boolean Networks [5, 6] is a discrete dynamic modeling approach. This technique is highly useful as it allow for a qualitative dynamic description of system behavior without the need for kinetic parameters. Boolean Networks are mainly used to model biological networks where biological components are defined as binary nodes and their regulations

are represented by directed edges. The state of a node in the network can be determined by another node thorough Boolean functions. Although Boolean models construction is simple, it leads to emergent dynamic behaviors in the model and provides important insights into biological systems. BooleanNet [20], BoolNet [21], SimBoolNet [22] and ChemChains [23] are existing software tools used for Boolean network modeling of biological systems.

Statecharts

Statecharts [7] is a computer visual language initially designed to study reactive systems, such as automotive and aerospace systems. Statecharts can be used to model natural and living systems thanks to structure definition allowing to use classes of objects that can be connected with each other to specify a precise state. Statecharts are modular and multi-level hierarchical models where one can modify objects, connections, states and transitions at any level. Using Statecharts it is possible to build any object from a lower scale object and observe the behavior at any level. This formalism is has a graphical representation and is most suitable for multi-scale modeling. It can be executed and simulated with a complete mathematically precise dynamic semantics. The most relevant tool for Statecharts in systems biology is Rhapsody [24]. This tool allows models to be fully executable with the ability to display Statecharts in a useful way. In addition to that, modelers can check the consistency of a chart against the model itself.

Rule based Systems

Rule-based modeling [8, 9] is a well-known framework in biology systems modeling thanks to its notation which is quite similar to the representation of chemical

reactions in biological systems. Also, it is considered a good approach to the problem of multi-state components in biological models. The main property of this modeling technique is that rules are independent and can be modified easily. The reactions rules are defined as high-level transformations of classes of species into biochemical networks. This technique is user-friendly because the syntax of the model can be saved in human readable text and visualized using graphical representations. BIOCHAM [25] is a rule-based tool for modeling biochemical systems with unique features for developing and correcting models.

Process Algebras

Process algebra [8, 9] is a family of formal languages for modeling concurrent systems including systems biology where biological species can be modeled as processes. This technique is able to provide formal specification without any uncertainty about the interactions and communications between concurrent processes. The application of process algebra in systems biology is largely focused on signaling pathways. Beta Workbench [26] is a scalable modeling tool based on process algebra that has been developed for biology from the very beginning. The main goal of this tool is to facilitate modeling of biological systems at different levels of abstractions, providing various features such as typed and dynamically varying interfaces of biological components, sensitivity-based interaction, spatial information, and hybrid parameters specification.

Stochastic models

Stochastic models [4] is another approach used in biology modeling, such that the state of a system is describe by a vector, where each one of its elements represents the number of molecules of a specific species at a certain time. The state of the system can

evolve based on the sequence of reactions. The sequence and time of reactions are determined probabilistically. Therefore, a stochastic differential equation is used to capture the probability distribution over the system states at each time point.

Most modeling approaches can be adapted to be used to study stochastic effects in different biological systems, such as stochastic Boolean networks and stochastic Petri Nets [27]. Different languages are used to describe the stochastic dynamics of systems. For instance, k language [4] is used to describe large biopathways. For slightly smaller systems, one may choose stochastic simulations where Gillespie's algorithm and its variants are often used [4].

Stochastic simulations are useful to study the behavior of a system if there is a lack of knowledge. The overall behavior of the system can be analyzed after running a large number of simulations [27]. Authors in [28] performed a comparison between a deterministic and stochastic simulation of a Petri net model that was originally presented expressed as a system of differential equations.

The Petri Nets formalism

The Petri Net formalism [10] is another powerful modeling tool that is extensively used in the computational biology field. This formalism is used to model concurrent activities and will be discussed in detail in the background section.

There are many databases that store biopathways in different formats to be used as inputs for Petri Net-based tools to perform modeling and simulation. Formats that are often mentioned in the literature include:

- System Biology Markup Language (SBML) [18].

- BioModels Database [29, 30] with various formats such as SBML, BioPax [31] and XPP [32, 33].
- Standard Petri Net Markup Language (PNML) [34].
- Integrated databases to a unified format [35].
- Data format translators, such as KEGG translator [16].
- Blenx: Simulation programming language systems [36].

Tools such as Snoopy [37, 38] provides translation from SBML to different extensions of Petri Nets in a snoopy-based XML format. Also, Cell Illustrator [39] translates SBML and BioPax formats to CSML [39], which is an XML-based format for Cell Illustrator tool. The MPath2PN application [40] applies an automatic transformation of metabolic pathway models in KGML format [35] into Petri Nets markup language (PNML). PNML [34] is the standard format for Petri Nets, so it can be used as an input for many Petri Nets modeling and simulation tools. The source of the metabolic pathways is publicly available databases such as KEGG [41]. The translation is implemented using extensible Stylesheet Language Transformation (XSLT).

1.2 Transformation from Differential Equations to Petri Nets

Many models in system biology are represented with ordinary differential equations (ODE), which require kinetic information in order to perform different analysis. This mathematical representation provides accurate and quantitative results. However, Petri Nets have an advantage over ordinary differential equations for several reasons. For instance, the Petri Net formalism can provide more quantitative and qualitative information than complex ODE models, while ordinary differential equations are restricted to modeling

dynamic changes only. Additionally, Petri Nets provide a graphical and human readable representation of the system being modeled, which helps in monitoring the model dynamic behaviors and observe its structural characteristics. Finally, Petri Nets are able to easily model both deterministic and stochastic processes [42]. [43] Explains in more details how Petri Nets are more expressive than ODE.

Authors in [33] proposed a translation from a system of ODE to a Continuous Petri Nets (CPN) model which is represented by a markup language called Abstract Petri Net Notation (APNN) [44]. The uniqueness of this transformation is based on conditions that have been taken into account to derive an algorithm that ensures the existence of only one Continuous Petri Net representation for each system of ordinary differential equations. This uniqueness allows for a conversion in both directions. Models created using this translation are suitable for quantitative and stochastic analysis.

Another study in [43] proposed a translation from ordinary differential equations to Timed Continuous Petri Net (TCPN) with product semantics [45]. The first step of translation is shifting the ODE behavior to the first orthant, then translating each term of the differential equation into TCPN elements taking into account semantic connections.

In [42], authors illustrated the power of Petri Nets in a bone remodeling system as an event-driven tool where it is possible to dynamically and graphically represent a system of discrete interactions and behaviors. Actual bone remodeling processes in living tissues are represented using discrete events, as opposed to ODE which can only be useful for modeling continuous time frame interactions.

For some cases and for some specific models, ODE models can be preferred for observing some behaviors and performing quantitative analysis. For instance, an ODE

system is a natural choice for quantitative modeling of biochemical networks. In [14], authors integrated Petri Nets and ODE models for a deep analysis of large models. This integration is achieved by deriving the continuous ordinary differential equations model from the analysis of a discrete Petri Net model of a specific system.

2. Background

2.1 Petri Nets Modeling

The Petri Net formalism [10] is a formal method used to model and analyze complex concurrent systems and offering an elegant graphical representation. A Petri Net can be seen as a bipartite graph with two types of nodes: places and transitions, connected by directed arcs. The role of places is to hold tokens (data such as species) that can be consumed or produced based on the firing of transitions. Tokens are produced in a place when an input transition is fired, and consumed when an output transition is fired. A transition represents an event that may occur in the modeled system. Petri Nets provides a natural framework in which quantitative and qualitative analysis are integrated in one formalism.

The behavior of a Petri Net is controlled by the firing rules of its transitions. The firing of a transition depends on the marking of its pre-places. If all pre-places are sufficiently marked, a transition is enabled and can be fired. Both the firing of a transition and the movement of tokens from one place to another change the marking of connected places. The behavior of a net is determined by the continuous firing of transitions [37].

Petri Nets [46] combine the following interesting features:

Readability: Petri Nets representation is easily comprehensible. Although it does allow unambiguousness, it still provides clear and readable details at different levels of abstractions with different resolutions of details.

Executability: Petri Nets can be executed with any suitable tool such as Snoopy [43, 44], in order to provide a comprehensive visualization that allows users to study the behavior of a system.

Causality: The structure of connections between Petri Net's elements define the causality relationship among its different components.

Static analysis: Petri Nets allow for a variety of static analysis techniques such as liveness [11] and boundedness [11].

The use of Petri Nets in biology was suggested for the first time in [47] to qualitatively analyze metabolic pathways. The Petri Net extension that was proposed could be used to model biological processes. Some of the currently available Petri Nets extensions are: colored, timed, stochastic, continuous, hybrid, extended hybrid, functional and hybrid functional. Each extension is defined as follows:

Colored Petri Nets: This extension [11] can be used to model large systems using different categories of tokens called *Colors*. With this extension it is possible to represent different dynamic behaviors modeled as different colors in the same model.

Timed Petri Nets: Time Petri Nets [46] are characterized by transitions associated with firing delays. This extension is suitable to model real time systems.

Stochastic Petri Nets: This extension [48] was proposed to model stochastic systems in biology in order to reduce the model implementation delays. In stochastic Petri Nets, the firing of a transition is not instantaneous. A random delay is computed based on

a probabilistic distribution. This delay is interpreted as the reaction rate. Stochastic Petri nets is mainly used when quantities of molecules are represented using discrete amount [11].

Continuous Petri Nets: This extension [11] places and transitions handle continuous quantities instead of discrete tokens. Continuous places are marked with real numbers called marks, while continuous transitions are associated with variables called speed. A transition's speed defines the rate of quantity transformation.

Hybrid Petri Nets: In hybrid Petri Nets [11], discrete and continuous processes can be combined within one model. Continuous elements model continuous changes, while discrete elements model discrete changes often associated with delays. This extension is well suited to model complex biological systems and processes.

Extended hybrid Petri Nets: This modeling extension [49] includes new stochastic transitions in addition to discrete and continuous transitions, and can be used to model and simulate different systems, including biological processes.

Functional Petri Nets: The idea behind this extension [46] is to use state-dependent functions to dynamically adjust the weight of arcs. This supports the idea of self-modifying nets.

Hybrid Functional Petri Nets: This type of Petri Nets [11] is developed to help model biological processes more accurately. New types of arcs are introduced, such as inhibitory and test arcs. Many biological systems have been modeled and simulated using this powerful combination of hybrid and functional Petri features. Genomic Object Net [50, 51] is a commercial bio-simulation software tool developed for biologists based on Hybrid Functional Petri Net architecture.

Many other tools based on the various Petri Net extensions also exist. For instance, Snoopy [37, 38] is a unified Petri Net tool to design, simulate and animate tokens interactions. Models created can be hierarchically structured to simplify building large models. This framework comprises different Petri Net classes, such as qualitative, continuous, stochastic and hybrid Petri Nets. These different models can be converted into each other. The three main features of Snoopy are extensibility, adaptability, and platform independence. Another tool is SimHPN [52] which is a MATLAB embedded package for hybrid Petri Nets that offers various simulations and analysis capabilities.

2.2 Why Petri Nets?

The main power of Petri Nets is the ability to represent concurrent processes in a simple way. This powerful technique encompasses nearly all other formalisms (qualitative, quantitative, discrete, continuous, deterministic or stochastic). Furthermore, a major reason for selecting Petri Net as a modeling approach in our work is its expressiveness and graphical representation [49]. In fact, Petri Nets have the capability to integrate qualitative and quantitative analysis, and offer various features useful to model biological systems and processes such as executability, causality and static analysis. In the field of computational biology, Petri Net components can be easily mapped to the components of a biochemical network. Molecular species (metabolites, enzymes) can be represented by places where the input of each place corresponds to a reactant and the place's output identifies the reaction products. Petri Nets transitions are mapped chemical reactions, while arcs connecting places with transitions with a token value that indicates the stoichiometry reactions. The number of tokens of each place

indicates the amount of substance of each species [40]. This resemblance between Petri Nets and chemical reactions makes Petri Nets a well-known formalism in the computational biology field.

Chapter 3:

1. Methodology

The main purpose of this work is to develop an automatic smart translation of a biopathway process modeled as a system of ordinary differential equations to an equivalent Hybrid Functional Petri Net model, to take advantage of available simulation tools as well as taking advantage of the better expressiveness of the Petri Net formalism. In this chapter, the implementation phase is divided into sections wherein the techniques and steps applied are demonstrated in details. The biological models we will be considering are taken from an online databases of biopathways available for researchers to uses for free, and stored in XPP files format [53].

1.1 Data Source: BioModels Database

Among the various existing databases of biological models, BioModels database [29, 30] is selected as the data source for this work. This database is a web-based repository of computational models of different categories of biological processes such as metabolic process, cellular process, immune system process, biological regulation, multi-organism process and others. These models are either published in the literature or automatically generated from pathway resources such as KEGG [42]. Models are categorized into curated and non-curated models. Curated models are manually curated by checking the structure and semantic of each model. However, the semantics of non-curated published models are not verified yet.

Different Formats

Under the curated models category, each model is available in different file formats such as SBML, BioPAX, Scilab, octave/MATLAB file, VCML, and XPP. The reason behind choosing this database, is the availability of the XPP format [53] we use in our work. This format is generated from SBML models and represents ordinary differential equations (ODE) systems. In the evaluation stage of our work, the MATALB format is used to validate our approach by comparing the simulation of a model in MATLAB with the simulation of its translation from the XPP format to Petri Nets.

XPP Format: ODE files

In its structure, an XPP file is composed of series of lines representing declarations of parameters, formulas such as ordinary differential equations and auxiliary quantities. The syntax of XPP is described in [53].

1.2 Our translation approach from ODE system to HFPN

A Hybrid Functional Petri Nets Petri Net can be used to model a biopathway process in an intuitive way with places representing variables of the systems while transitions represent equations that describe how these variables change over time.

The HFPN formalism is already known to be more expressive than differential equations systems. As a result, various biological systems and processes have already been successfully modeled and simulated using HFPNs such as genetic regulation, metabolic networks and transduction signal system [11].

Authors in [54] transformed two neurobiological models represented by a system of ordinary differential equations into a Hybrid functional Petri Nets representation. The first model is a phenomenological synapse model and the second one is a molecular-level model of the CaMKII regulation pathway. The translation was performed manually using an ad-hoc approach by mapping ordinary differential equations Hybrid Functional Petri Nets elements. Figure 1 shows the phenomenological synapse model with both representations: Ordinary differential equations (left) and Hybrid Functional Petri Nets (right).

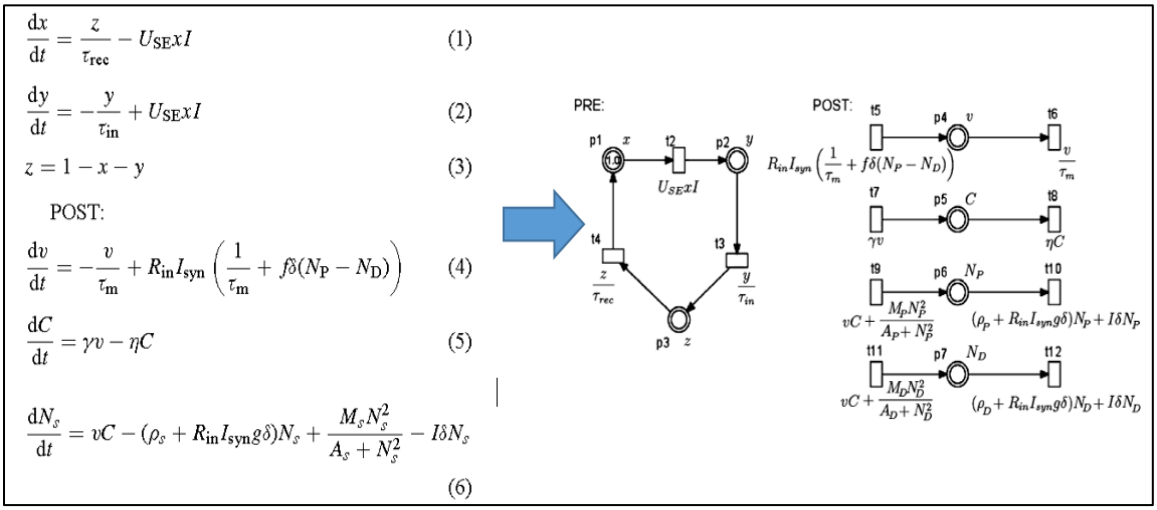

Figure 1. Translation of differential equations into a Hybrid Functional Petri Net representation of the phenomenological synapse model.

Naïve translating from a system of ODEs to a HFPN

From Figure 1 we can see that the translation from the system of ordinary differential equations representing the phenomenological synapse model to a Hybrid Functional Petri Net can be conducted as follows:

- Each variables appearing on the left-hand side of each differential equation is mapped to a continuous place.
- The right -hand side of any differential equation is processed as follows:
  - Each positive term is mapped to the speed of production of an input transition.
  - Each negative term is mapped to the speed of consumption of an output transition.

From these observations we conclude that the system of differential equations should be first put in the general *Sum of Products* form represented in equation (1).

$$\frac{dx_i}{dt} = \sum_{j=1}^{n}(c_{ij}\, x_j) \; + g_i \quad for\ (i = 1..n) \qquad (1)$$

Where c_{ij} (*i=1..n, j=1..n)* are constant coefficients (possibly null), g_i (*i=1..n)* are constants (possibly null) and *n* is the number of variables (or equations) in the system.

In this form, the right hand side of each differential equation is composed of a sum of terms with positive and negative coefficients. If we separate positive and negative terms, the system can be rewritten as follows:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n}(a_{ij}\, x_j) - \sum_{j=1}^{n}\left(b_{ij}\, x_j\right) + \; g_i \quad for\ (i = 1..n) \qquad (2)$$

Where a_{ij} and b_{ij} are positive constant coefficients (possibly null).

Now, given a system of ODEs in a *Sum of Products* form, our translation approach consists in mapping each variable x_i appearing in the left hand side of each differential

equation to a place with a similar name in the HFPN translation. The right hand side positive terms (including constant g_i if it is positive) will be mapped to input transitions to that place, while negative terms (including constant g_i if it is negative) will be mapped to output transitions from that place. Null terms are ignored off course. The speed of each transition is set to be equal the corresponding term. Figure 2 gives a representation of such a transformation for the differential equation:

$$\frac{dx_i}{dt} = \sum_{j=1}^{n}(a_{ij}\, x_j) - \sum_{j=1}^{n}(b_{ij}\, x_j) + g_i \qquad (3)$$

Where g_i is positive.

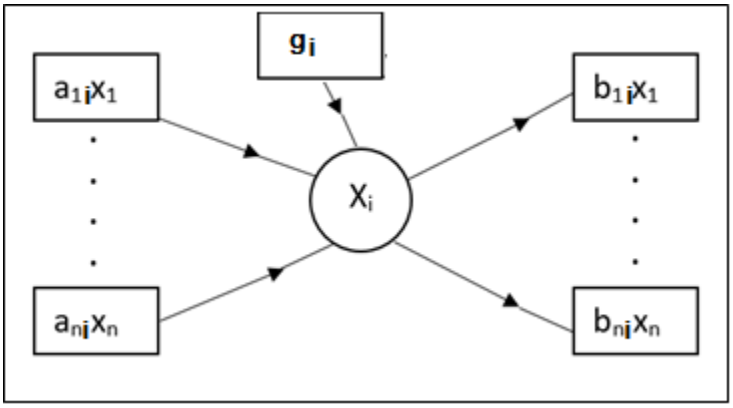

Figure 2. Petri Net representation of Equation (3)

Establishing smart Connections

If each differential equation is translated according to steps described above, the resulting HFPN would be semantically equivalent to the original system of differential equations, but will consist of many disconnected components not related to each other.

While this could be fine for simulation purposes, it still misses one very important aspect which is: human readability. In fact, with disconnected components it is very difficult to understand how the modeled system works so as to draw conclusion and extend the model. Therefore, we propose an additional step to connect related transitions to reflect the functioning of the modeled system. This step, consists simply in merging any output transition with any input transition if they both have the same speed equation.

As an example, consider the following system of ODEs:

$$\frac{dx_0}{dt} = (a_1 \times x_1) - (b_1 \times x_0)$$

$$\frac{dx_1}{dt} = (b_1 \times x_0) - (b_1 \times x_1)$$

After translation we end up with a HFPN with two places x_0 and x_1, each one of them having one input transition and one output transition. Given that the output transition of x0 has the same speed equation as the input transition of x1, then these two transitions can be merged into a single transition (see figure 3). The impact of this merge on the meaning of the network is very important as it allow us to understand that whatever is produced in pace x_0 is consumed and used to fill place x_1.

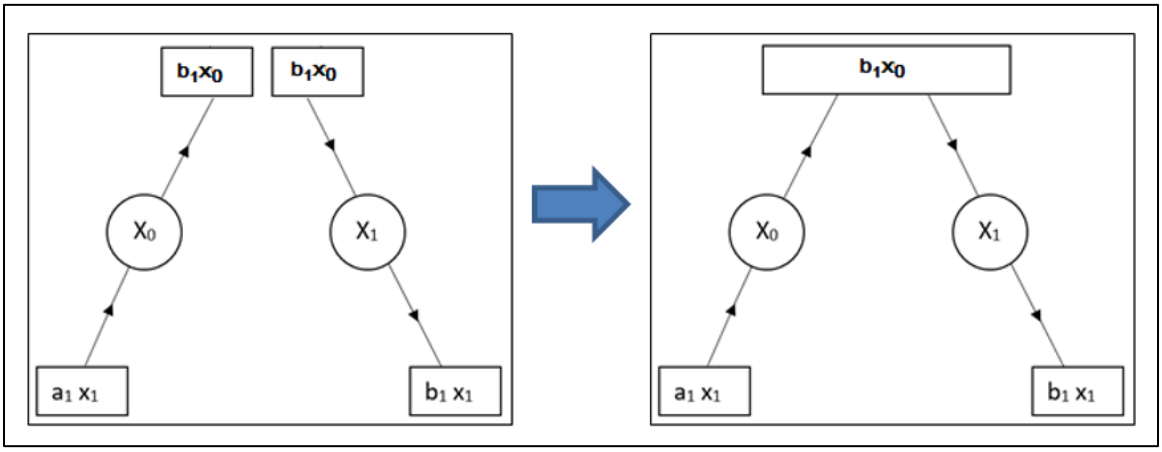

Figure 3. Smart Connection of Petri Net Elements

Consequently, the obtained model can interpret much better the biopathway in a more meaningful representation where the relation between different places (molecules and species) is captured through smart connections.

2. Implementation

In the following subsections, the steps taken in order to process an XPP file representing a biopathway modeled as a system of ODEs are described, with an example in details.

2.1 From XPP file to abstract syntax tree

Models we translate are systems of ordinary differential equations representing biopathways and stored in XPP files. Figures 4, 5, and 6 show a full XPP file of the Cell Cycle model. The file starts with comments where each commented line in the file starts

with '#'. The file combines parameters declaration, variables definition, and ordinary differential equations. The end of the file is indicated by the keyword 'done'.

```
# Model name = Tyson1991 - Cell Cycle 6 var
# is urn:miriam:biomodels.db:MODEL6614644188
# is urn:miriam:biomodels.db:BIOMD0000000005
# isDescribedBy urn:miriam:pubmed:1831270
# some function definitions that are allowed in SBML but not valid in xpp
ceil(x)=flr(1+x)
@delay=50
# Compartment: id = cell, name = cell, constant
par cell=1.0
# assignmentRule: variable = YT
YT=Y+YP+M+pM
aux YT=YT
# assignmentRule: variable = CT
CT=C2+CP+M+pM
aux CT=CT
# Reaction: id = Reaction1, name = cyclin_cdc2k dissociation
# Local Parameter:   id =  k6, name = k6
par k6=1.0
Reaction1=cell*k6*M
# Reaction: id = Reaction2, name = cdc2k phosphorylation
# Local Parameter:   id =  k8notP, name = k8notP
par k8notP=1000000.0
Reaction2=cell*C2*k8notP
# Reaction: id = Reaction3, name = cdc2k dephosphorylation
# Local Parameter:   id =  k9, name = k9
par k9=1000.0
Reaction3=cell*CP*k9
# Reaction: id = Reaction4, name = cyclin cdc2k-p association
# Local Parameter:   id =  k3, name = k3
par k3=200.0
```

Figure 4. The XPP file of the Cell Cycle model (part 1)

```
Reaction4=cell*CP*k3*Y
# Reaction: id = Reaction5, name = deactivation of cdc2 kinase
# Local Parameter:   id =  k5notP, name = k5notP
par k5notP=0.0
Reaction5=cell*k5notP*M
# Reaction: id = Reaction6, name = cyclin biosynthesis
# Local Parameter:   id =  k1aa, name = k1aa
par k1aa=0.015
Reaction6=cell*k1aa
# Reaction: id = Reaction7, name = default degradation of cyclin
# Local Parameter:   id =  k2, name = k2
par k2=0.0
Reaction7=cell*k2*Y
# Reaction: id = Reaction8, name = cdc2 kinase triggered degration of cyclin
# Local Parameter:   id =  k7, name = k7
par k7=0.6
Reaction8=cell*k7*YP
# Reaction: id = Reaction9, name = activation of cdc2 kinase
# Local Parameter:   id =  k4, name = k4
par k4=180.0
# Local Parameter:   id =  k4prime, name = k4prime
par k4prime=0.018
Reaction9=cell*pM*(k4prime+k4*(M/CT)^2)
# Species:   id = EmptySet, name = EmptySet
par EmptySet=0.0
aux EmptySet=EmptySet
```

Figure 5. The XPP file of the Cell Cycle model (part 2)

```
# Species:   id = C2, name = cdc2k, affected by kineticLaw
init C2=0.0
dC2/dt=(1/(cell))*(( 1.0 * Reaction1) + (-1.0 * Reaction2) + ( 1.0 * Reaction3))
# Species:   id = CP, name = cdc2k-P, affected by kineticLaw
init CP=0.75
dCP/dt=(1/(cell))*(( 1.0 * Reaction2) + (-1.0 * Reaction3) + (-1.0 * Reaction4))
# Species:   id = M, name = p-cyclin_cdc2, affected by kineticLaw
init M=0.0
dM/dt=(1/(cell))*((-1.0 * Reaction1) + (-1.0 * Reaction5) + ( 1.0 * Reaction9))
# Species:   id = pM, name = p-cyclin_cdc2-p, affected by kineticLaw
init pM=0.25
dpM/dt=(1/(cell))*(( 1.0 * Reaction4) + ( 1.0 * Reaction5) + (-1.0 * Reaction9))
# Species:   id = Y, name = cyclin, affected by kineticLaw
init Y=0.0
dY/dt=(1/(cell))*((-1.0 * Reaction4) + ( 1.0 * Reaction6) + (-1.0 * Reaction7))
# Species:   id = YP, name = p-cyclin, affected by kineticLaw
init YP=0.0
dYP/dt=(1/(cell))*(( 1.0 * Reaction1) + (-1.0 * Reaction8))
# Species:   id = YT, name = total_cyclin, involved in a rule
# Species:   id = CT, name = total_cdc2, involved in a rule
@ meth=cvode, tol=1e-6, atol=1e-8
# @ maxstor=1e6
@ bound=40000, total=200
done
```

Figure 6. The XPP file of the Cell Cycle model (part 3)

XPP files are first parsed using a parser we wrote in JavaCC following a set of rules corresponding to the grammar rules that define the syntax of XPP format. The parser generates an abstract syntax tree (AST) following the steps shown in figure 7.

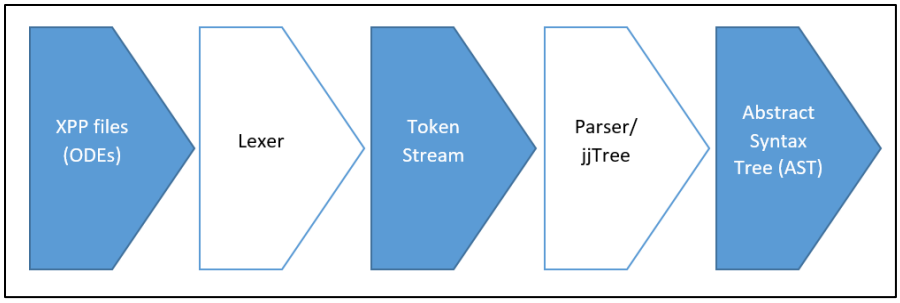

Figure 7. XPP parsing steps

Each node in the tree has a specific number of children defined in the parser rules. Each child node represents a specific element or set of elements in the file such as variables, equations, differential equations or constants. The structure of the generated abstract syntax tree and its child nodes are shown in figure 8, 9, 10, 11 and 12.

Figure 8 shows the first level of nodes of the abstract syntax tree, where ASTstart is the root of the tree. The children of the root are:

- ASTparameter node: represents parameters declarations.
- ASToptionalParameter node: represents optional parameters which start with symbol '@'.
- ASTlinearEquation node: represents variables definitions in terms of linear equations.
- ASTdiffrentialEquation node: represents the ordinary differential equations.

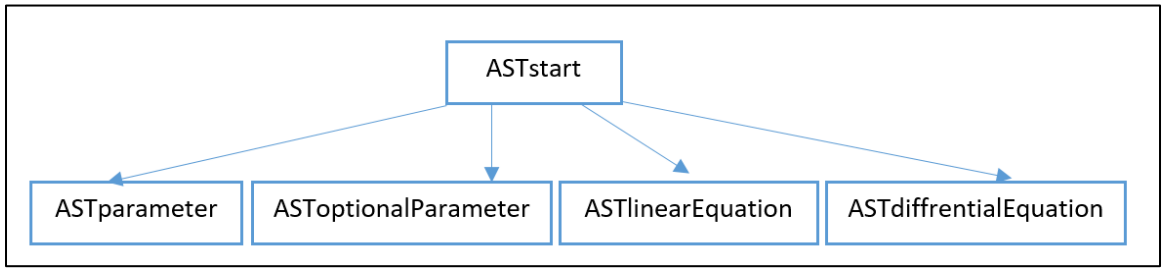

Figure 8. First level nodes of the generated Abstract Syntax Tree

Figure 9 shows child nodes of the ASTparameter node which are: ASTexpression and ASTsymbol. The ASTexpression node refers to a term that combines symbols and operators where ASTsymbol node defines constants and variables.

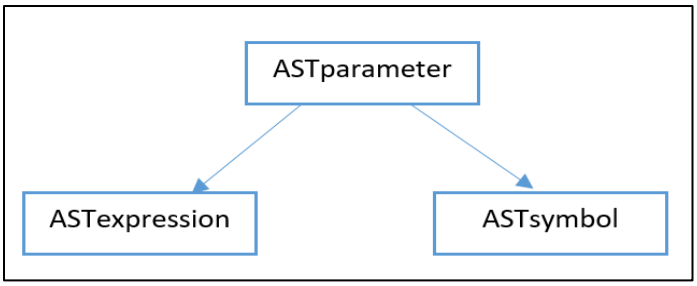

Figure 9. ASTparameter node with two children nodes

Figure 10 represents the ASToptionalParameter node which has two children nodes that are identical to ASTparameter node's children. The only difference between the two nodes is the occurrence of symbol '@' in the optional parameter.

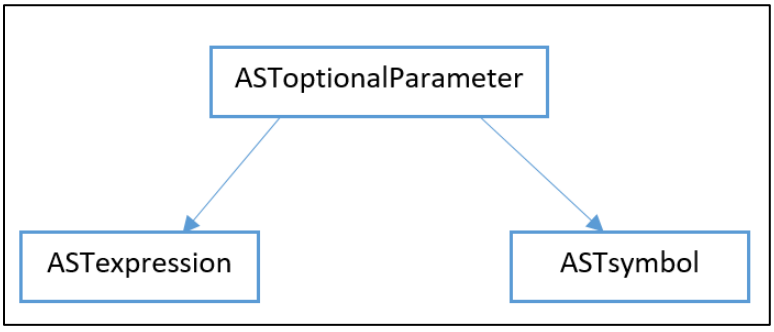

Figure 10. ASToptionalParameter node with two children nodes

Figure 11 shows ASTlinearEquation node's children which combines linear expressions, symbols, as well as left and right parentheses.

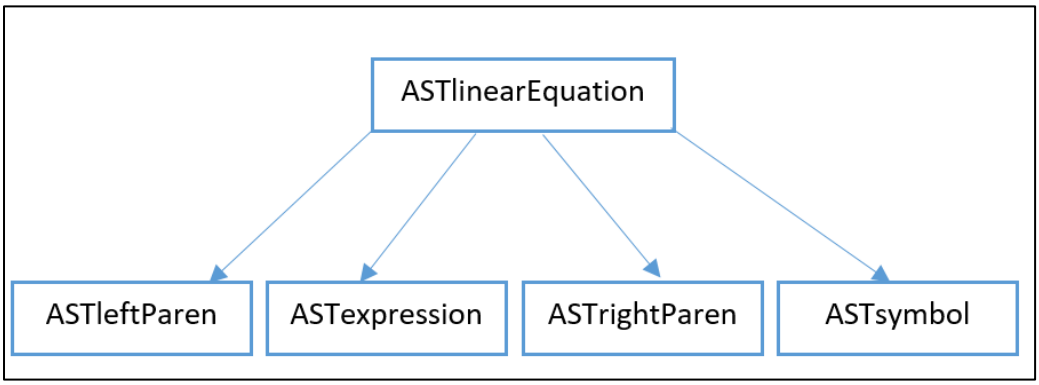

Figure 11. ASTlinearEquation node with four children nodes

Figure 12 shows ASTdiffrentialEquation node's children. The ASTdiffrentialEquation node has a single child which is ASTdiffexp that represents the right-hand side of the differential equation. The ASTdiffexp node is defined by three child nodes which are: ASTdiffterm, ASTplusOp and ASTminusOp. The ASTdiffterm node can be an ASTdiffexp node along with left and right parentheses or an ASTsymbol with division or time operators.

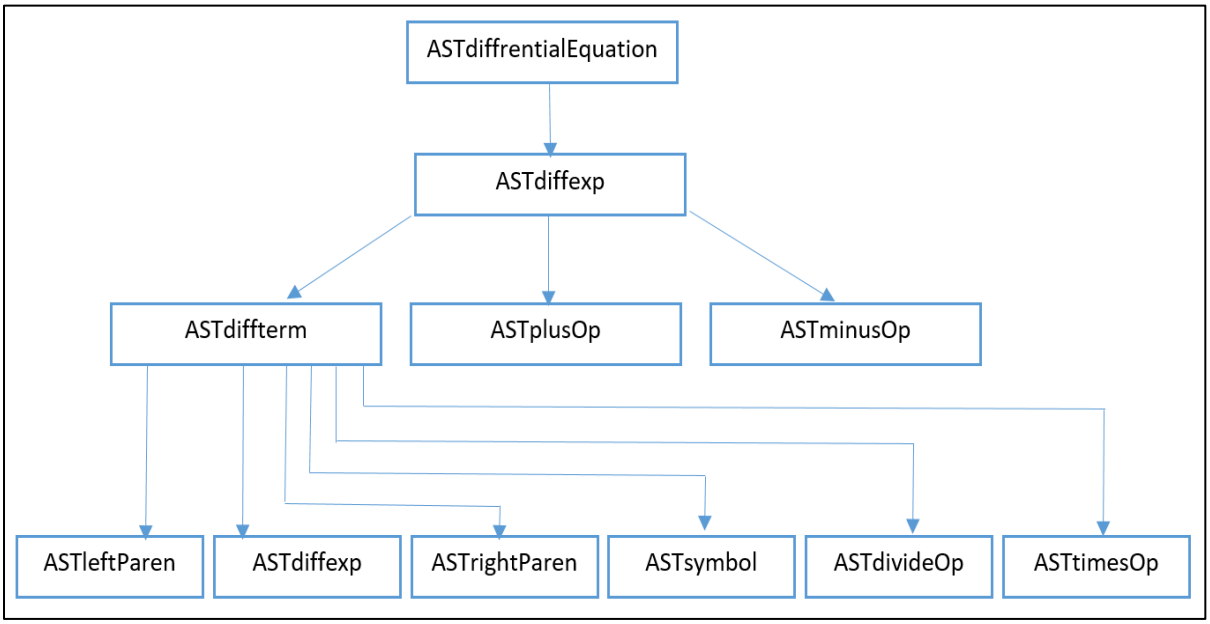

Figure 12. ASTdiffrentialEquation node with multi-level children nodes

2.2 Tree Processing

To translate a system of ordinary differential equations to a HFPN, each one of its equations needs first to be put into the form of sum of products. This operation is performed on the generated parse tree, because traversing the multi-levels nodes of the generated abstract syntax tree makes accessing the different parts of each ordinary differential equation easy and simple.

Substituting variables in each ODE term

In an XPP file only variables appearing on the right hand side of differential equations are translated into places in the resulting HFPN. These variables are called place variables. All other variables and constants need to be substituted with their

corresponding equations and values appearing at the beginning of the XPP files. At the end of this process, only place variables are left, while all other variables are substituted.

Figure 13 shows an ordinary differential equation extracted from BIOMD01 XPP file. Figure 14 presents the declarations and constants values of each variable that exists in the ODE shown in figure 13.

```
init BLL=0.0
dBLL/dt=(1/(comp1))*(( 1.0 * React1) + (-1.0 * React2))
```

Figure 13. An ordinary differential equation extracted from BIOMD01 XPP file

```
par comp1=1.0E-16

par kf_1=1500.0

par kr_1=16000.0

par kf_2=30000.0

par kr_2=700.0

React1=comp1*(kf_1*BL - kr_1*BLL)

React2=comp1*(kf_2*BLL - kr_2*ALL)
```

Figure 14. Variables declaration

After applying the substitution of variables and constants values in the ordinary differential equation shown in figure 13, the ODE takes the form shown in Equation (4).

$$\frac{dBLL}{dt} = \left(\frac{1}{(1.0E-16)}\right) \times \left(\left(1.0 \times \left(1.0E - 16 \times (1500.0 \times BL - 16000.0 \times BLL)\right)\right) -\right.$$

$$\left(1.0 \times \left(1.0E-16 \times (30000.0 \times BLL - 700.0 \times ALL)\right)\right) \quad (4)$$

Applying distributive law in each ODE

Most of ordinary differential equations in XPP files are represented in the form of a term multiplied by an expression of a sum of products. To put each ODE in the form of a sum of products, the distributive law is applied. The right hand side of the differential equation takes the form represented in Equation (5).The application of the distributive law on each ODE is achieved by walking the abstract syntax tree nodes up and down in order to rewrite the equations in the desired form.

$$\frac{dBLL}{dt} = \left(\frac{1}{(comp1)}\right) \times (1.0 \times React1) + \left(\frac{1}{(comp1)}\right) \times (-1.0 \times React2) \quad (5)$$

Forming positive terms

The last step of data processing is expressing each ODE with positive variables and constants. The negative sign of each variable/constant is taken out of the whole term to get positive constants and variables in each single term as exemplified in Equation (6).

$$\frac{dBLL}{dt} = \left(\frac{1}{(comp1)}\right) \times (1.0 \times React1) - \left(\frac{1}{(comp1)}\right) \times (1.0 \times React2) \quad (6)$$

The final presentation of the ordinary differential equation after applying variables substitution and distributive law, as well as forming positive terms is shown in Equation (7).

$$\frac{dBLL}{dt} = \times \left(\left(\frac{1}{(1.0E-16)} \right) \times \left(1.0 \times \left(1.0E - 16 \times (1500.0 \times BL - 16000.0 \times BLL) \right) \right) \right) - \left(\left(\frac{1}{(1.0E-16)} \right) \times \left(1.0 \times \left(1.0E - 16 \times (30000.0 \times BLL - 700.0 \times ALL) \right) \right) \right) \quad (7)$$

2.3 Translation from the abstract syntax tree to a HFPN

The conversion from ODE representation into Hybrid Functional Petri Nets is straight forward. First, each place variable is translated into a place with a similar name. After that, each positive term appearing on right hand side of a differential equation is translated to an input transition to the place corresponding to the place variable appearing on the left hand side tor the differential equation. The speed of that transition is set to be the term itself. The same is done for each negative term, except that the corresponding transition is an output transition to the place.

By running this process on the ordinary differential equation given in Equation (7), the following actions are taken to construct the corresponding Hybrid Functional Petri Net model.

- Create place BLL corresponding to place variable BLL.
- Create input transition to place BLL with speed: $\left(\frac{1}{(1.0E-16)} \right) \times \left(1.0 \times \left(1.0E - 16 \times (1500.0 \times BL - 16000.0 \times BLL) \right) \right)$.
- Create output transition of place BLL with speed: $\left(\frac{1}{(1.0E-16)} \right) \times \left(1.0 \times \left(1.0E - 16 \times (30000.0 \times BLL - 700.0 \times ALL) \right) \right)$.

Figure 15 shows the created Petri Nets part that corresponds the single ordinary differential equation given in Equation (7).

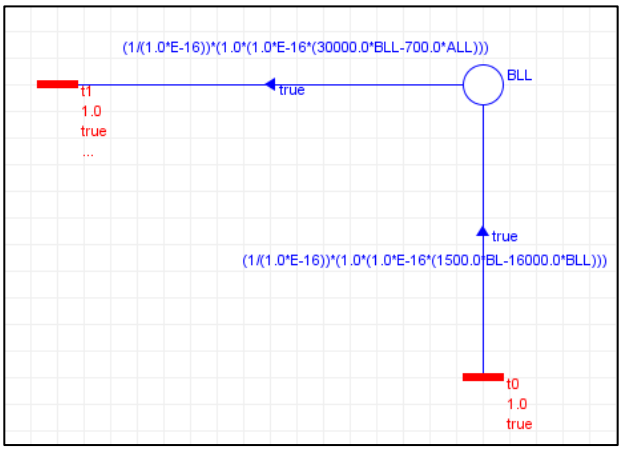

Figure 15. HFPN of place BLL and its input and output transitions

To illustrate establishing smart connections between transitions, let us consider three places BLL, BL, and ALL represented by the three differential equations shown in figure 16. These equations are extracted from BIOMD01 XPP file.

```
dBLL/dt=(1/(comp1))*(( 1.0 * React1) + (-1.0 * React2))

dBL/dt=(1/(comp1))*(( 1.0 * React0) + (-1.0 * React1) + (-1.0 * React6))

dALL/dt=(1/(comp1))*(( 1.0 * React2) + ( 1.0 * React4) + (-1.0 * React11))
```

Figure 16. Three ordinary differential equations extracted from BIOMD01 XPP file

Figure 17 shows the naïve translation from ordinary differential equations shown in figure 16 into Hybrid Functional Petri Nets. Each place is connected to its related

transitions, while common transitions appear more than once in the network (e.g. transitions t_0 and t_1).

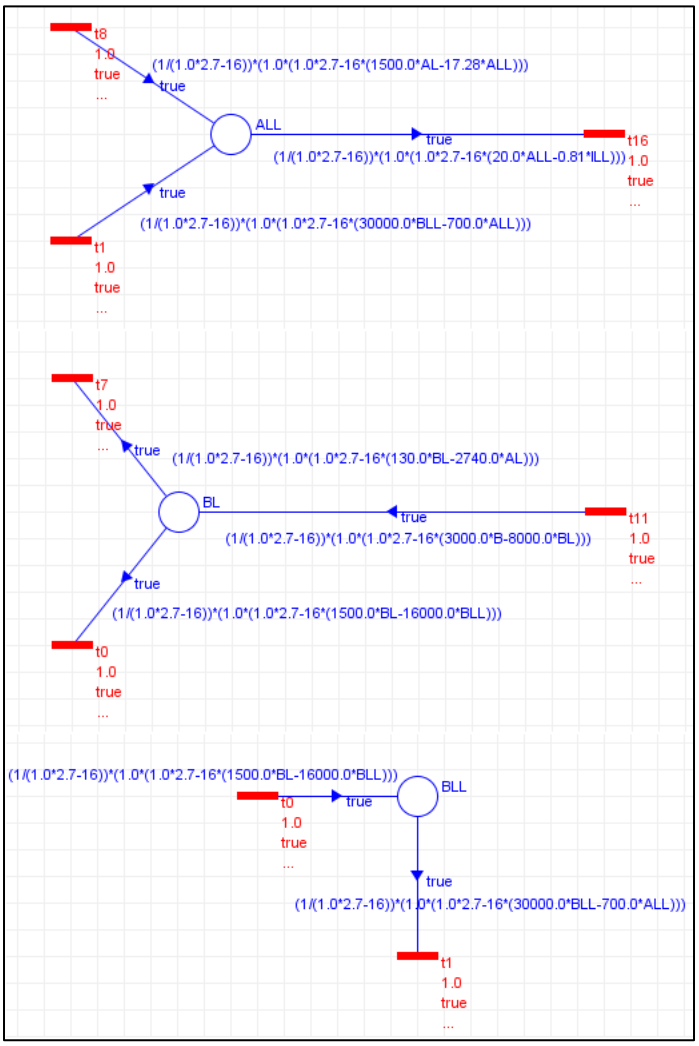

Figure 17. Naive translation of three ODEs into HFPN model

After establishing the smart connections, we end up with the more meaningful Petri Net illustrated in figure 18. In this Petri net, it is clearer how the different components of the models system are related to each other and how they communicate with each other.

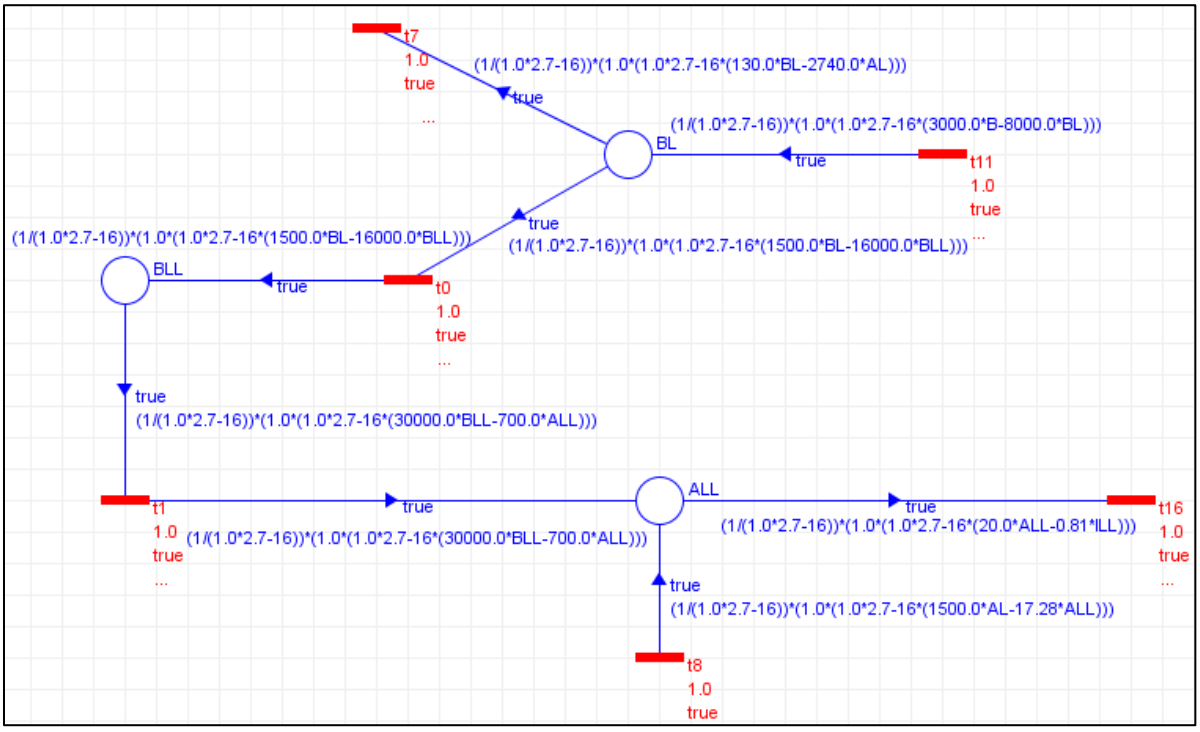

Figure 18. HFPN with smart connections between places BL, BLL, and ALL

2.4 Extension of the tool: Real time Studio

To validate our translation approach, we implemented it as an extension to the tool Real Time Studio [12]. This tool is a Petri Nets integrated environment used for modeling, simulation and automatic verification of real-time systems. In its current version, the tool supports two Petri Nets extensions: Interpreted Time Petri Nets (ITPN) [12] and Hybrid Functional Petri Nets (HFPN) [11]. Our extension to the tool enables it to read XPP files and translate them automatically to Hybrid Functional Petri Nets. Input files should follow the XPP format available on BioModels database in order to be parsed successfully.

Chapter 4:

1. Results

1.1 Hybrid Functional Petri Net Model

The approach described in the methodology part is used to generate Hybrid Functional Petri Nets from biopathway models under the curated category of BioModels database, and stored in XPP files.

Figure 19 shows the Petri Net model of the biopathway: *Nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ* [55].

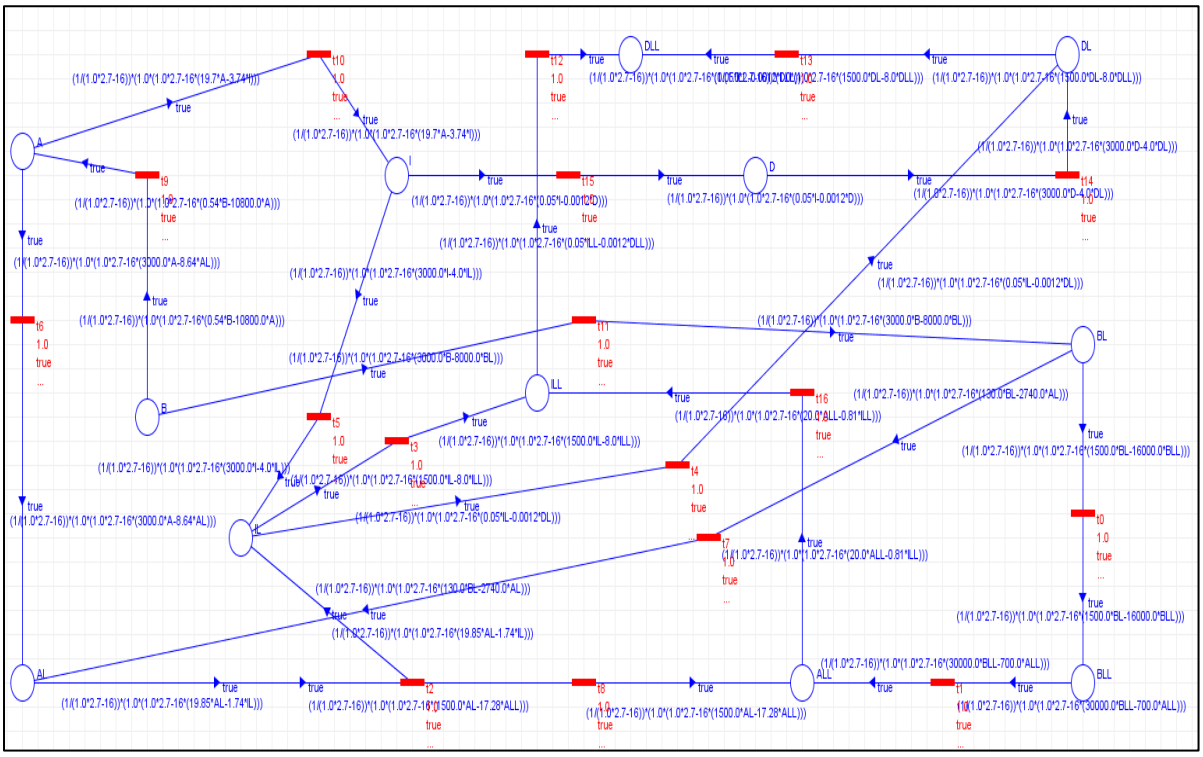

Figure 19. Hybrid Functional Petri Net model of the biopathway nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ

The HFPN in figure 19 consists of 12 places and 17 transitions which represent 12 ordinary differential equations (see figure 20). Each equation is composed of positive and negative terms that refer to input and output transitions. This model shows different places having more than two input/output transitions thanks to the use of smart connections, which reduced the number of transitions from 34 to 17, while adding more meaning to the network.

Equations in figure 20 represent the ordinary differential equations of the biopathway nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ. All these equations are linear and all constant values are defined and declared at the beginning of the ODE/XPP file.

```
dBLL/dt=(1/(comp1))*(( 1.0 * (comp1*(kf_1*BL-kr_1*BLL))) + (-1.0 * (comp1*(kf_2*BLL-kr_2*ALL))))

dIL/dt=(1/(comp1))*(( 1.0 * (comp1*(kf_7*I-kr_7*IL))) + (-1.0 * (comp1*(kf_8*IL-kr_8*ILL))) + ( 1.0 * (comp1*(kf_10*AL-kr_10*IL))) + (-1.0 * (comp1*(kf_15*IL-kr_15*DL))))

dAL/dt=(1/(comp1))*(( 1.0 * (comp1*(kf_3*A-kr_3*AL))) + (-1.0 * (comp1*(kf_4*AL-kr_4*ALL))) + ( 1.0 * (comp1*(kf_6*BL-kr_6*AL))) + (-1.0 * (comp1*(kf_10*AL-kr_10*IL))))

dA/dt=(1/(comp1))*((-1.0 * (comp1*(kf_3*A-kr_3*AL))) + ( 1.0 * (comp1*(kf_5*B-kr_5*A))) + (-1.0 * (comp1*(kf_9*A-kr_9*I))))

dBL/dt= (1/ (comp1))*(( 1.0 * (comp1*(kf_0*B-kr_0*BL))) + (-1.0 * (comp1*(kf_1*BL-kr_1*BLL))) + (-1.0 * (comp1*(kf_6*BL-kr_6*AL))))

dB/dt=(1/(comp1))*((-1.0 * (comp1*(kf_0*B-kr_0*BL))) + (-1.0 * (comp1*(kf_5*B-kr_5*A))))

dDLL/dt=(1/(comp1))*(( 1.0 * (comp1*(kf_13*DL-kr_13*DLL))) + ( 1.0 * (comp1*(kf_16*ILL-kr_16*DLL))))

dD/dt=(1/(comp1))*((-1.0 * (comp1*(kf_12*D-kr_12*DL))) + ( 1.0 * (comp1*(kf_14*I-kr_14*D))))
```

dILL/dt=(1/(comp1))*((1.0 * (comp1*(kf_8*IL-kr_8*ILL))) + (1.0 * (comp1*(kf_11*ALL-kr_11*ILL))) + (-1.0 * (comp1*(kf_16*ILL-kr_16*DLL))))

dDL/dt=(1/(comp1))*((1.0 * (comp1*(kf_12*D-kr_12*DL))) + (-1.0 * (comp1*(kf_13*DL-kr_13*DLL))) + (1.0 * (comp1*(kf_15*IL-kr_15*DL))))

dI/dt=(1/(comp1))*((-1.0 * (comp1*(kf_7*I-kr_7*IL))) + (1.0 * (comp1*(kf_9*A-kr_9*I))) + (-1.0 * (comp1*(kf_14*I-kr_14*D))))

dALL/dt=(1/(comp1))*((1.0 * (comp1*(kf_2*BLL-kr_2*ALL))) + (1.0 * (comp1*(kf_4*AL-kr_4*ALL))) + (-1.0 * (comp1*(kf_11*ALL-kr_11*ILL))))

Figure 20. System of ODE modeling the nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ

2. Evaluation:

To validate our approach, we compared simulation results of original models written as systems of ODEs with their translations into HFPNs using our approach. The simulation of original models is performed using MATLAB, while HFPNS are simulated using the tool Real Time Studio. To illustrate our results, we give in what follows the results we obtained when simulating for models of the nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ shown in figure 19.

2.1 Real Time Studio Simulation Results

The HFPN model of the nicotinic Excitatory Post-Synaptic Potential in a Torpedo electric organ is first generated by translating its ODE model using our approach (see figure 19). Using a simulation time interval 0.00001 time units, simulation curves

representing how quantities associated with place variables change over time are captured during four different time periods: 0.005, 0.1, 0.5, and 1.0. The observation of simulation results for different time periods give us a deep understanding about the dynamic changes during the simulation process.

Figure 21 represents the first part of Real time Studio simulation results of 12 place variables during 0.005 time units, then, figure 23 shows the continuation of the simulation until reaching 0.1 time units. Figure 25 and 27 illustrate the obtained curves during 0.5 and 1.0 time units respectively.

2.2 Differential Equations Simulation Results

Ordinary differential equations are simulated using MATLAB. The MATLAB model's file of each simulated biological system is downloaded from the BioModels database. Similarly to Real Time Studio simulation, the outcome of ordinary differential equations simulation is represented by a graph that combines all places curves in one graph. The same simulation settings of RT Studio tool are applied to MATLAB simulation where the simulation interval is set to 0.00001 and the curves changes are captured during different time intervals: 0.001, 0.1, 0.5 and 1.0 as illustrated in figure 22, 24, 26, and 28 respectively.

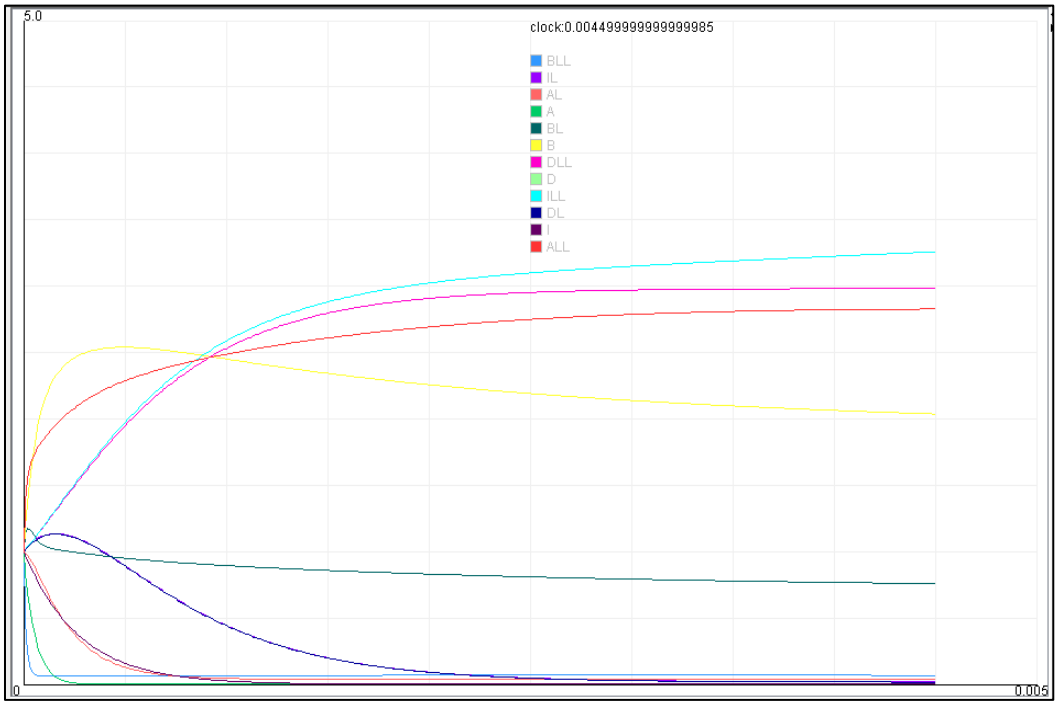

Figure 21. RT Studio Simulation during time unit 0.005

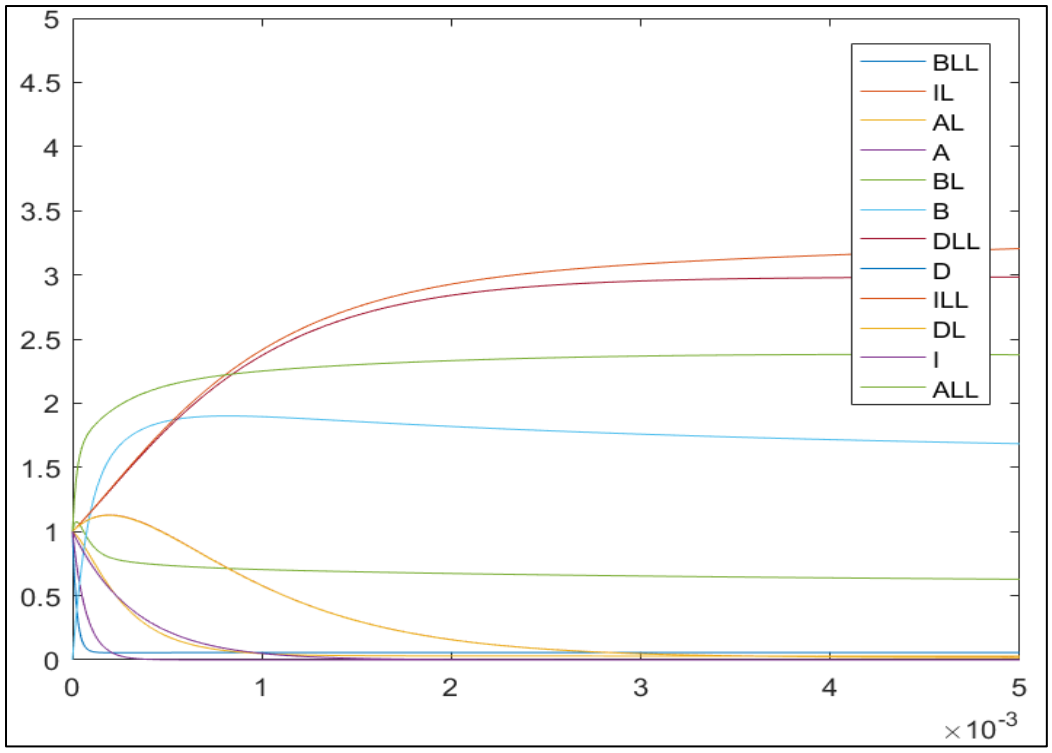

Figure 22. MATLAB Simulation during time unit 0.0

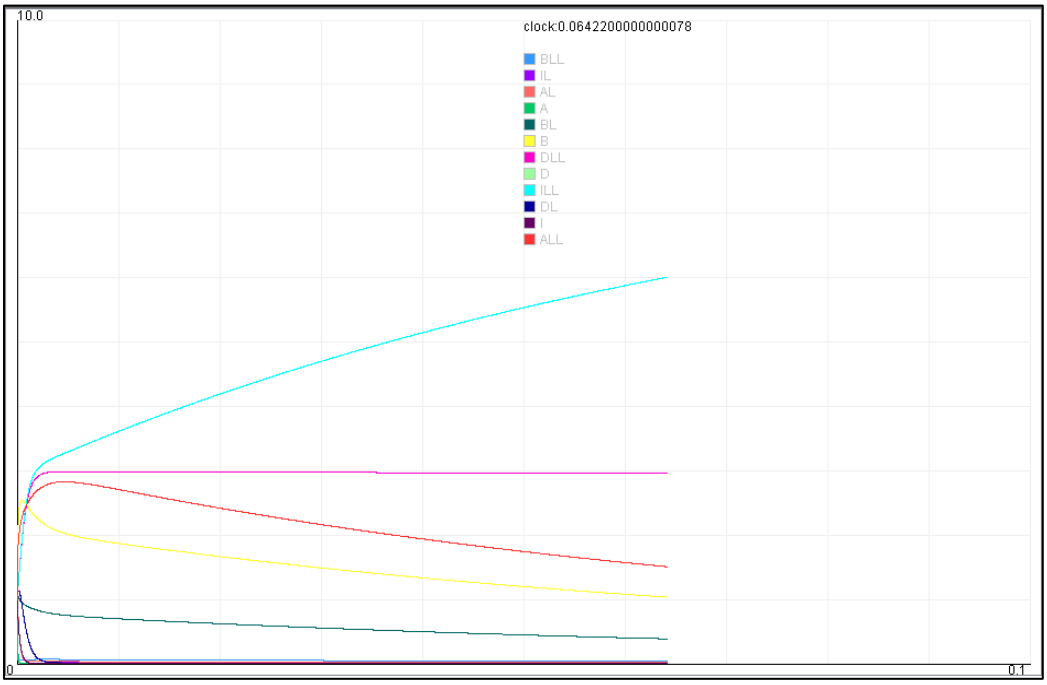

Figure 23. RT Studio Simulation during time unit 0.1

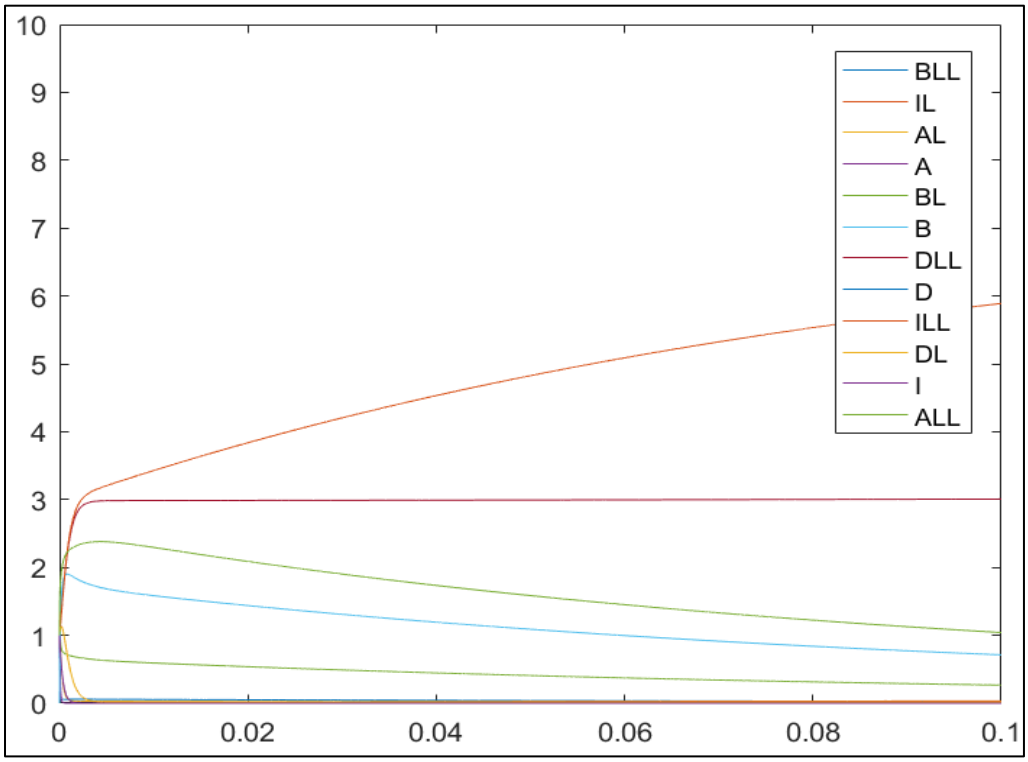

Figure 24. MATLAB Simulation during time unit 0.1

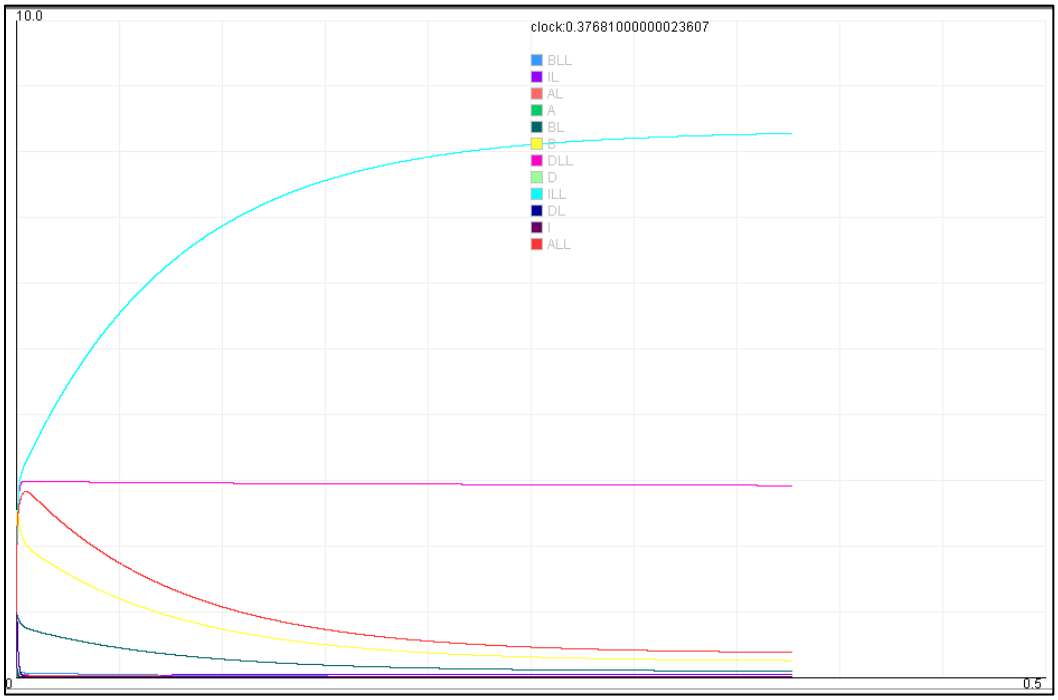

Figure 25. RT Studio Simulation during time unit 0.5

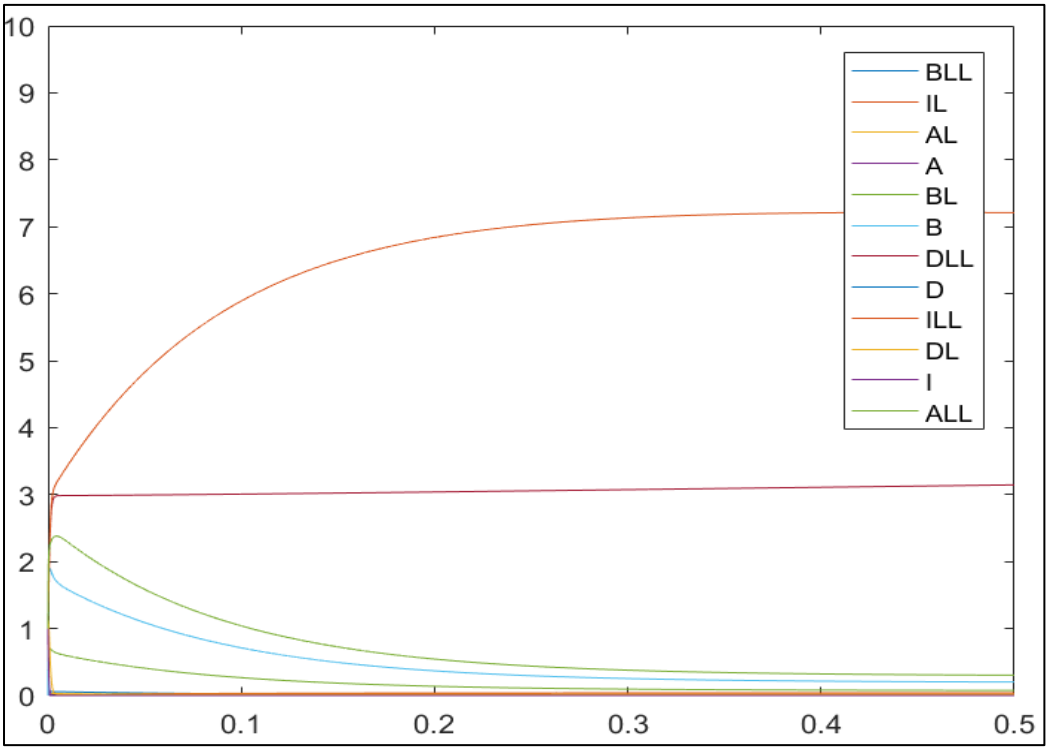

Figure 26. MATLAB Studio Simulation during time unit 0.5

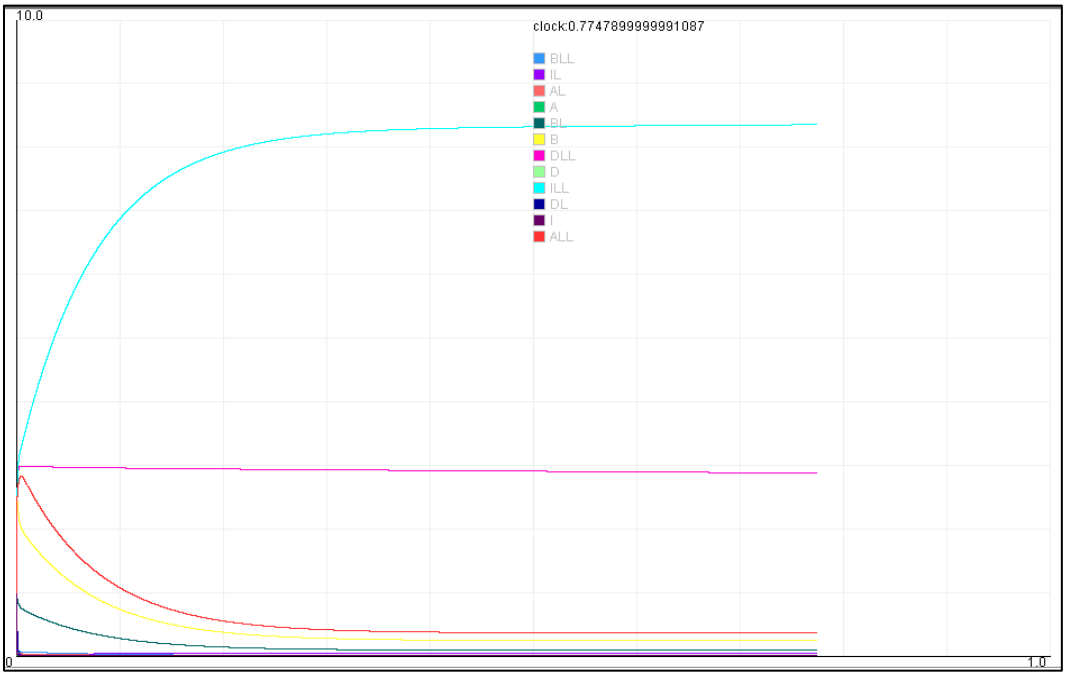

Figure 27. RT Studio Simulation during time unit 1.0

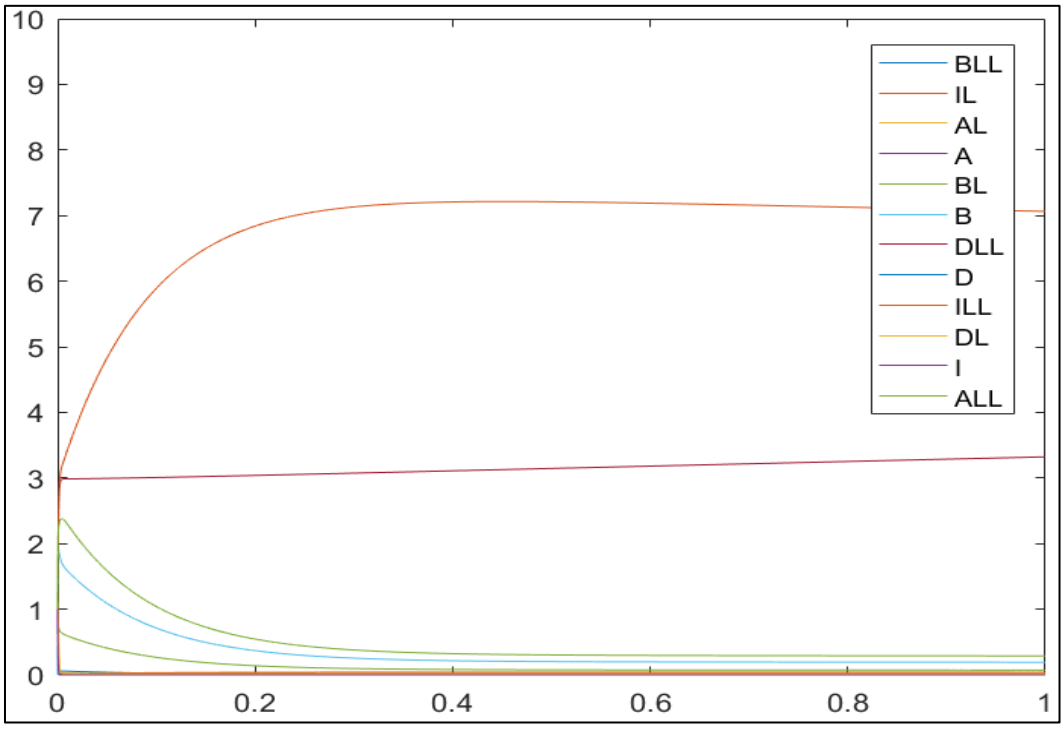

Figure 28. MATLAB Studio Simulation during time unit 1.0

Chapter 5:

1. Discussion

The comparison of the two simulation graphs at different time intervals confirms that the simulation results of both model's representations are similar with identical curves for all places variables. Each curve in the graph shows the dynamic changes of each place variable in the network corresponding to a specific molecule in the biopathway. As it was previously stated, Hybrid Functional Petri Nets representation are more expressive than ordinary differential equations system [56], therefore, it was anticipated to obtain the same simulation results for both model's representations.

The obtained results confirm that our proposed automatic conversion approach from systems of ODEs to Hybrid Functional Petri Nets is correct. Using our translation approach, it is now easier for researchers to visualize models of biopathways, which can provide them with better insights into their functioning. Additionally, our translation approach can be exploited in the field of teaching and learning since HFPNs are more intuitive to understand than systems of ODEs.

2. Future Directions

This work can be extended to allow conversion from other different representations of biopathways such as SBML and BioPax into Hybrid Functional Petri Nets. Additionally, our conversion approach can be applied to translate models not necessarily related to biology, other field where ODEs are used such as chemistry, physics, mechanics, electronics and power systems.

3. Conclusion

In this thesis, an automatic conversion approach is developed to transform biopathways presented by ordinary differential equations into Hybrid Functional Petri Nets model. The obtained Petri Net model is not only automatically generated, but it also preserves the semantics of the translated model. Due to the intuitive representation of Petri Nets, the obtained model is human readable. It facilitates capturing and understanding complex system behaviors and interactions which is not possible to capture with ordinary differential equations. Simulation results validate our approach which can open the door to it application in other fields where ODEs are used.

REFERENCES

[1]. Machado, D., Costa, R. S., Rocha, M., Rocha, I., Tidor, B., & Ferreira, E. C. (2009, June). A critical review on modelling formalisms and simulation tools in computational biosystems. In International Work-Conference on Artificial Neural Networks (pp. 1063-1070). Springer Berlin Heidelberg.

[2]. Chowbina, S. R., Wu, X., Zhang, F., Li, P. M., Pandey, R., Kasamsetty, H. N., & Chen, J. Y. (2009). HPD: an online integrated human pathway database enabling systems biology studies. BMC bioinformatics, 10(11), 1.

[3]. National Institutes of Health. (2000). NIH working definition of bioinformatics and computational biology. Biomedical Information Science and Technology Initiative Committee (BISTIC), 1.

[4]. Liu, B., & Thiagarajan, P. S. (2012). Modeling and analysis of biopathways dynamics. Journal of Bioinformatics and Computational Biology, 10(04), 1231001.

[5]. Wang, R. S., Saadatpour, A., & Albert, R. (2012). Boolean modeling in systems biology: an overview of methodology and applications. Physical biology, 9(5), 055001.

[6]. Thakar, J., & Albert, R. (2010). Boolean models of within-host immune interactions. Current opinion in microbiology, 13(3), 377-381.

[7]. Cohen, I. R., & Harel, D. (2007). Explaining a complex living system: dynamics, multi-scaling and emergence. Journal of the royal society interface, 4(13), 175-182.

[8]. Bartocci, E., & Lió, P. (2016). Computational Modeling, Formal Analysis, and Tools for Systems Biology. PLoS Comput Biol, 12(1), e1004591.

[9]. Machado, D., Costa, R. S., Rocha, M., Ferreira, E. C., Tidor, B., & Rocha, I. (2011). Modeling formalisms in systems biology. AMB express, 1(1), 1.

[10]. Rozenberg, G., & Reisig, W. (1998). Lectures on Petri Nets I: Basic Models. Springer Lecture Notes in Computer Science, 1491.

[11]. Hardy, S., & Robillard, P. N. (2004). Modeling and simulation of molecular biology systems using petri nets: modeling goals of various approaches. Journal of bioinformatics and computational biology, 2(04), 619-637.

[12]. Hadjidj, R., & Boucheneb, H. (2013). RT-Studio: A Tool for Modular Design and Analysis of Realtime Systems Using Interpreted Time Petri Nets. In PNSE+ ModPE (pp. 247-254).

[13]. Motta, S., & Pappalardo, F. (2013). Mathematical modeling of biological systems. Briefings in Bioinformatics, 14(4), 411-422.

[14]. Gilbert, D., & Heiner, M. (2006, June). From Petri nets to differential equations–an integrative approach for biochemical network analysis. In International Conference on Application and Theory of Petri Nets (pp. 181-200). Springer Berlin Heidelberg.

[15]. Van Gend, C., & Kummer, U. (2001). STODE-automatic stochastic simulation of systems described by differential equations. In Proceedings of the 2nd International Conference on Systems Biology (pp. 326-333). Chicago

[16]. Tomita, M., Hashimoto, K., Takahashi, K., Shimizu, T. S., Matsuzaki, Y., Miyoshi, F., & Hutchison, C. A. (1999). E-CELL: software environment for whole-cell simulation. Bioinformatics, 15(1), 72-84.

[17]. Fages, F., Gay, S., & Soliman, S. (2015). Inferring reaction systems from ordinary differential equations. Theoretical Computer Science, 599, 64-78.

[18]. Hucka, M., Finney, A., Sauro, H. M., Bolouri, H., Doyle, J. C., Kitano, H., & Cuellar, A. A. (2003). The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. Bioinformatics, 19(4), 524-531.

[19]. Fages, F., Gay, S., & Soliman, S. (2012). Automatic curation of SBML models based on their ODE semantics (Doctoral dissertation, INRIA).

[20]. Albert, I., Thakar, J., Li, S., Zhang, R., & Albert, R. (2008). Boolean network simulations for life scientists. Source code for biology and medicine, 3(1), 1.

[21]. Müssel, C., Hopfensitz, M., & Kestler, H. A. (2010). BoolNet—an R package for generation, reconstruction and analysis of Boolean networks. Bioinformatics, 26(10), 1378-1380.

[22]. Zheng, J., Zhang, D., Przytycki, P. F., Zielinski, R., Capala, J., & Przytycka, T. M. (2010). SimBoolNet—a Cytoscape plugin for dynamic simulation of signaling networks. Bioinformatics, 26(1), 141-142.

[23]. Helikar, T., & Rogers, J. A. (2009). ChemChains: a platform for simulation and analysis of biochemical networks aimed to laboratory scientists. BMC systems biology, 3(1), 1.

[24]. Harel, D., & Gery, E. (1996, May). Executable object modeling with statecharts. In Proceedings of the 18th international conference on Software engineering (pp. 246-257). IEEE Computer Society.

[25]. Soliman, S., & CONTRAINTES, I. R. P. (2009, February). Modelling biochemical reaction networks with biocham extracting qualitative and quantitative information from the structure. In Proceedings of the 6th Vienna Conference on Mathematical Modelling MATHMOD (Vol. 9, pp. 2304-2312).

[26]. Dematté, L., Priami, C., & Romanel, A. (2008). The Beta Workbench: a computational tool to study the dynamics of biological systems. Briefings in Bioinformatics, 9(5), 437-449.

[27]. Schlitt, T. (2013). Approaches to modeling gene regulatory networks: a gentle introduction. In Silico Systems Biology, 13-35.

[28]. Castaldi, D., Maccagnola, D., Mari, D., & Archetti, F. (2012, August). Stochastic simulation of the coagulation cascade: a petri net based approach. In European Conference on Parallel Processing (pp. 248-262). Springer Berlin Heidelberg.

[29]. BioModels Database. http://www.ebi.ac.uk/biomodels.

[30]. Le Novere, N., Bornstein, B., Broicher, A., Courtot, M., Donizelli, M., Dharuri, H., & Snoep, J. L. (2006). BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. Nucleic acids research, 34(suppl 1), D689-D691.

[31]. BioPAX: Biological Pathway Exchange. http://www.biopax.org/index.php

[32]. XPP Tutorial: Basic Idea and Introduction, http://www.math.pitt.edu/~bard/bardware/tut/

[33]. Soliman, S., & Heiner, M. (2010). A unique transformation from ordinary differential equations to reaction networks. PloS one, 5(12), e14284.

[34]. Billington, J., Christensen, S., Van Hee, K., Kindler, E., Kummer, O., Petrucci, L., & Weber, M. (2003, June). The Petri net markup language: concepts, technology, and tools. In International Conference on Application and Theory of Petri Nets (pp. 483-505). Springer Berlin Heidelberg.

[35]. Kanaris, I., Moutselos, K., Chatziioannou, A., Maglogiannis, I., & Kolisis, F. N. (2008, October). Building in-silico pathway SBML models from heterogeneous sources. In BioInformatics and BioEngineering, 2008. BIBE 2008. 8th IEEE International Conference on (pp. 1-6). IEEE.

[36]. Dematté, L., Priami, C., & Romanel, A. (2008, June). The BlenX language: a tutorial. In International School on Formal Methods for the Design of Computer, Communication and Software Systems (pp. 313-365). Springer Berlin Heidelberg.

[37]. Blätke, M., Heiner, M., & Marwan, W. (2011). Tutorial—Petri Nets in Systems Biology. Otto von Guericke University Magdeburg, Magdeburg Centre for Systems Biology.

[38]. Rohr, C., Marwan, W., & Heiner, M. (2010). Snoopy—a unifying Petri net framework to investigate biomolecular networks. Bioinformatics, 26(7), 974-975.

[39]. Nagasaki, M., Doi, A., Matsuno, H., & Miyano, S. (2005). Petri net based description and modeling of biological pathways. Algebraic Biology, 15(1), 19-31.

[40]. Baldan, P., Cocco, N., & Simeoni, M. (2013). Representing and Comparing Metabolic Pathways as Petri Nets with MPath2PN and CoMeta. Electronic Notes in Theoretical Computer Science, 299, 5-13.

[41]. Ogata, H., Goto, S., Sato, K., Fujibuchi, W., Bono, H., & Kanehisa, M. (1999). KEGG: Kyoto encyclopedia of genes and genomes. Nucleic acids research, 27(1), 29-34.

[42]. Li, L., & Yokota, H. (2009). Application of petri nets in bone remodeling. Gene regulation and systems biology 3.

[43]. Navarro-Gutiérrez, M., Ramírez-Treviño, A., & Gómez-Gutiérrez, D. (2013, September). Modelling the behaviour of a class of dynamical systems with Continuous

Petri Nets. In 2013 IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA) (pp. 1-6). IEEE.

[44]. Bause, F., Kemper, P., & Kritzinger, P. (1994). Abstract Petri net notation. Technical report, Dekanat Informatik, Univ.

[45]. Alla, H., & David, R. (1998). A modelling and analysis tool for discrete events systems: continuous Petri net. Performance Evaluation, 33(3), 175-199.

[46]. Heiner, M., & Gilbert, D. (2011, June). How might Petri nets enhance your systems biology toolkit. In International Conference on Application and Theory of Petri Nets and Concurrency (pp. 17-37). Springer Berlin Heidelberg.

[47]. Reddy, V. N., Mavrovouniotis, M. L., & Liebman, M. N. (1993, July). Petri net representations in metabolic pathways. In ISMB (Vol. 93, pp. 328-336).

[48]. Marsan, M. A., Balbo, G., Chiola, G., Conte, G., Donatelli, S., & Franceschinis, G. (1991). An introduction to generalized stochastic Petri nets. Microelectronics Reliability, 31(4), 699-725.

[49]. Proß, S., Bachmann, B., Janowski, S. J., & Hofestädt, R. (2012, December). A new object-oriented Petri net simulation Environment Based On Modelica. In Proceedings of the Winter Simulation Conference (p. 300). Winter Simulation Conference.

[50]. Nagasaki, M., Fujita, S., Matsuno, H., & Miyano, S. (2002). Genomic Object Net: II. Modelling biopathways by hybrid functional Petri net with extension. Applied Bioinformatics, 2(3), 185-188.

[51]. Cell Illustrator, http://www.cellillustrator.com/publications

[52]. Júlvez, J., Mahulea, C., & Vázquez, C. R. (2012). SimHPN: A MATLAB toolbox for simulation, analysis and design with hybrid Petri nets. Nonlinear Analysis: Hybrid Systems, 6(2), 806-817.

[53]. Format of ODE Files and Examples.

http://www.math.pitt.edu/~bard/bardware/tut/newstyle.html#newstyle

[54]. Hardy, S., & Robillard, P. N. (2005). Phenomenological and molecular-level Petri net modeling and simulation of long-term potentiation. Biosystems, 82(1), 26-38.

[55]. Edelstein, S. J., Schaad, O., Henry, E., Bertrand, D., & Changeux, J. P. (1996). A kinetic mechanism for nicotinic acetylcholine receptors based on multiple allosteric transitions. Biological cybernetics, 75(5), 361-379.

[56]. Matsuno, H., Doi, A., Nagasaki, M., & Miyano, S. (2000). Hybrid Petri net representation of gene regulatory network. In Pacific Symposium on Biocomputing (Vol. 5, No. 338-349, p. 87). Singapore: World Scientific Press.