Use of Discrete, Continuous and Hybrid Petri Nets in Bio-modelling

Mansurah Adam

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Applied Mathematics and Computer Science

Eastern Mediterranean University
January 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Assoc. Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Nazim Mahmudov
Chair, Department of Mathematics

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Applied Mathematics and Computer Science.

Prof. Dr. Rza Bashirov
Supervisor

Examining Committee

1. Prof. Dr. Rza Bashirov

2. Prof. Dr. Benedek Nagy

3. Assoc. Prof. Dr. Enver Ever

ABSTRACT

Petri net is a well-known field of mathematics and computer science, which has over the years been applied to problem solving in science and technology. Chemical processes, biomolecular systems, parallel and distributed computing, complex systems and the management of the flow of work are among areas where Petri nets has been successfully applied.

This thesis examines application of discrete, continuous and hybrid Petri nets in bio-modelling. Throughout the study, we examine interrelationship between discrete and continuous components, making sure of compatibility of component types. Three case studies examined are; the bottling system, production system, and olive soap production.

Keywords: Petri net, Bio-modelling, Discrete, Continuous, Hybrid, Simulation.

ÖZ

Petri ağları, bilim ve teknoloji alanlarında problem çözmede yıllardan beri uygulanmış bir matematik ve bilgisayar bilimi alanıdır. Petri ağlarının başarıyla uygulandığı alanlar arasında kimyasal reaksiyonlar, biyomoleküler sistemler, paralel hesaplamalar, karmaşık yönetim sistemleri yer almaktadır.

Bu tez, ayrık, sürekli ve hibrit Petri ağlarının biyo-modelleme uygulamalarını incelemektedir. Çalışma boyunca, ayrık ve sürekli bileşenler arasındaki ilişkiyi inceleyerek, bileşen türlerinin uyumluluğundan emin olduk. İncelenen üç çalışması; şişeleme sistemi, üretim sistemi ve zeytin sabunu üretimi.

Anahtar kelimeler: Petri net, biyo-modelleme, ayrık, sürekli, hibrid, simülasyon

DEDICATION

“If anyone travels on a road in search of knowledge, Allah will cause him to travel on one of the roads of paradise. He who issue forth in the search of knowledge is busy in the course of Allah till he returns from his quest. Seek knowledge from the cradle to the grave” Prophet Mohammed (SAW).

This quest and achievement is dedicated to Almighty Allah for the gift of life, health and knowledge for giving me love when there was doubt and faith when there was despair.

ACKNOWLEDGMENT

I cannot but thank God almighty for giving me the strength and knowledge to carry out this project.

I appreciate my supervisor and mentor, Prof. Dr. Rza Bashirov, for supervising this project and bringing the best out of me. Providing the best of advices at difficult times and encouraging me to do a worthwhile project. I came into this program under him at a time when I needed someone who will mentor and cover the gap in my academic pursuit, he decided to take up the responsibility. I remain grateful sir, for accepting to take me as your supervisee and student, and for making me to see and work on the research gaps. I thank you immensely for your tutelage and mentorship.

I also appreciate the efforts of the lecturers in the department of Mathematics, faculty of Arts and Sciences, Eastern Mediterranean University.

This acknowledgement will be incomplete if I do not appreciate my families who endured and sacrifice a lot for me during my MSc study.

I also appreciate the efforts of my colleagues in the department of Mathematics, and the University in general.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBRIEVATIONS

C-place	Continuous Place
CPN/ CP-Nets	Colored Petri Net
C-transition	Continuous transition
D-place	Discrete place
DPN	Differential Petri Nets
GSPN	Generalized Stochastic Petri Nets
GUI	Graphical User Interface
HLQPN	High Level Queuing Petri Nets
HPN	Hybrid Petri Nets
LLQPN	Low Level Queuing Petri Nets
P/T-net	Place Transition Net
QN	Queuing Nets
QPN	Queuing Petri Nets

Chapter 1

INTRODUCTION

Petri Nets, referred as place/transition nets (P/T-nets), is an attractive formalism which in recent years has gained acceptance in several fields such as computer programming, computer hardware, embedded systems, telecommunications, performance measurement, internetworking, biological systems, real-time systems, e-commerce, transportation system, operations research and office automation as a collaborative field of study.

Carl Adam Petri, the inventor of Petri Nets theory, introduced the field in his Ph.D. academic research in 1962. He started working on it 23 years earlier in order to describe chemical processes, producing a significant advancement in relevant areas, among which are the fields of parallel and distributed computing, complex systems, and management of the flow of work [1].

The Petri net formalism is ideal for naturally representation of molecular interactions between chemical substances and biological components. Its components may represent molecules and reactions, or even more complex entities and processes. This ability makes it extremely useful for signal transduction networks, gene regulatory networks and metabolic networks.

It is of practical interest to know how different structured net parts interact in a hybrid Petri net, meantime studying behavioral peculiarities like when a discrete marking expressed in tokens is transformed into a continuous one expressed in real numbers and vice versa.

We examined three case studies: Bottling system, Production System and Olive Soap Production System. We worked mainly with the Snoopy software [2]. We started with modelling of small Petri net fragment to appraise the functionality of the software. The three case studies were modeled, simulated and analyzed using Snoopy software.

It was observed that discrete place may enable or disable a continuous transition but firing action does not change its marking and vice versa.

A discrete Petri net in this context consists of discrete place and discrete transition with the markings represented by dots referred to as tokens which take on positive integer values. The markings of a continuous place, on the other hand, are assigned real number values instead of integers, which can be less than one. Hybrid Petri net is a combination of both discrete and continuous Petri net fragments. Furthermore, a discrete marking can be converted into a continuous marking and vice versa.

Different software tools used for Petri net simulation were examined and evaluated, each with its limitations and strengths. They are: SimHPN, HPsim, HiQPN tools, PetriSim, HISIm, Snoopy. Because of its ability to work with continuous, discrete and hybrid net fragments we use Snoopy for modeling and simulation of the case studies considered in this study.

This study aims at studying and analyzing the inter-relationship between descrete and continuous components, its places and transitions.

Chapter 2 gives an insight of what Petri net entails such as regular and high-level Petri nets. In this chapter, properties of Petri net and analysis methods are discussed. Chapter 3 mainly focuses on quantitative modelling with Petri net, looking at the review of existing software tools with emphasis on Snoopy software as a simulation tool. In Chapter 4, we examined three case studies including: Bottling System, Production System, and Olive Soap Production System.

Chapter 2

PETRI NETS

Petri nets is a graphical modeling framework which can be used in various areas [1]. Petri nets can also be used as a visual communication aid. As a graphical instrument, it facilitates to visualize change of a system's state by involving token game. On the other hand, when used as a mathematical tool, it is possible to construct algebraic equations, and other mathematical models determining the behavior of underlying dynamical systems. It is, therefore, safe to say that Petri nets can be utilized by practitioners and theoreticians, providing a strong medium of communication between them, since practitioners can learn from theoreticians on how to make their models methodically sound, and the theoreticians can also learn on how to make models more profound and realistic [3]. Petri Net can be used as visual communication which helps with flow charts, networks and block diagrams. To simulate the simultaneous and dynamic processes of a system, tokens are used [4].

2.1 Regular Petri Nets

Regular Petri net can be referred to as discrete Petri net. It is a directed, bipartite and weighted graph which consist of two kinds of nodes: places and transitions. Arcs connect a transition to a place or a place to a transition, but not both. The basic components of Petri net is illustrated in figure 1.

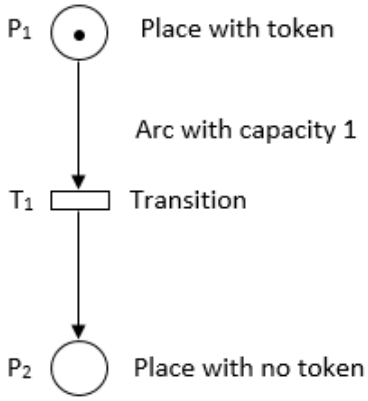

Figure 1: Basic components of a Petri net

Place: This is represented in our study by a circle.

Transition: Each transition is represented by a rectangular shape. **Source transition** has no input place while **sink transition** has no output place.

A transition is said to be **enabled** when the token in each of its input place is at least as much as the weight of the arc from that place to the transition. When fired, the tokens in the input places are moved to the output places. The enabled transition may or may not fire. Example of an enabled transition is shown in figure 2 below.

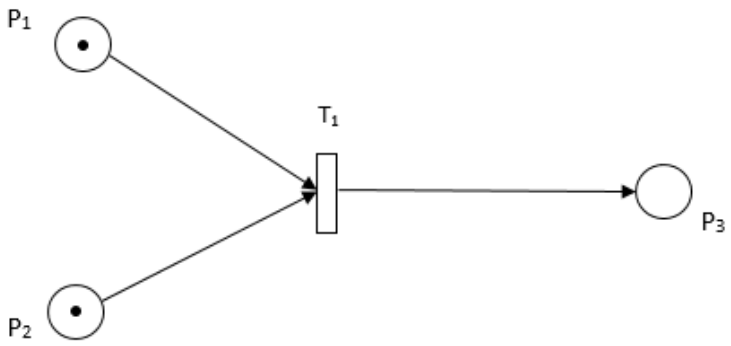

Figure 2: A Petri net with enabled transition (before firing)

Firing action removes tokens from input places and adds tokens to output places in accordance with the arc weights. This results in a new making of the net which is shown in figure 3.

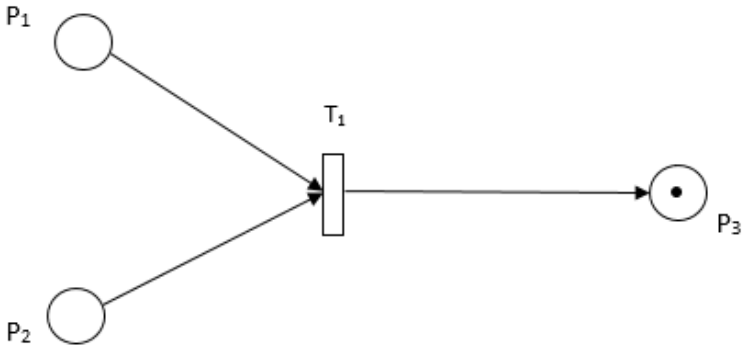

Figure 3: A disabled transition (after firing action)

A transition is **disabled** if the number of tokens in the input place is less than the input arc weight.

Arcs and Arc weight: A connection is made between places (which are finite but non-zero) and transitions with the use of an arc. These described places have tokens at times, and the state of the Petri net is the allocation of a token, or some tokens to a place. Arc weight (multiplicity) is the value given to arc, if not stated, it is assumed to be one by default.

Tokens: Tokens are represented as non-negative integers. It is denoted by m_i where i is the number identifying the place in question. The vector of all these individual markings composes the net marking for the Petri Net thus, $m = (m_1 \dots m_i)$ which defines the state of the system is described with the Petri Net.

A Petri net is a 5-tuple $PN = < P, T, A, W, M_0 >$ where: finite set of directed arcs.

$P = \{P_1, P_2, \dots P_m\}$ is a finite set of places,

$T = \{t_1, t_2, \dots t_n\}$ is a finite set of transitions,

$A \subseteq (P\ x\ T) \cup (T\ x\ P)$: *is* the set of arcs (flow relation),

$W: A \rightarrow \{1,2,3, \dots\}$ is a weight function,

$M_0: P \rightarrow \{1,2,3, \dots\}$ is the initial marking,

$P \cap T = \emptyset\ and\ P \cup T = \emptyset$

2.2 High-Level Petri Nets

High-level Petri nets are often used for modelling and simulation of engineering and scientific problems that have complex structures. It is an enhancement of the classical or regular Petri net. Regular Petri net can be considered with various extensions to expand its modelling ability. Colored Petri nets, timed Petri nets, stochastic Petri nets, continuous Petri nets, hybrid Petri nets are just few examples of high level Petri nets.

2.2.1 Colored Petri Nets

In colored Petri Nets (CP-Nets or CPNs), tokens or marks are assigned colors, thus forming a special classification of Petri nets with poor intuitive perception if compared with the generalized or finite capacity Petri nets. Each token is associated with attributes called colors. They provide robust interfaces for the modelling of complicated systems. It has an appealing graphical representation which makes it effortless to see the basic structure of a complex CPN model.

It worths noting that this form of Petri net can be transformed into ordinary Petri nets in case they have a finite number of colors. Such an example is found in Figure 4. It can be observed that the colored and ordinary Petri net possess the same structure. The structure can also be re-represented as a single one by adding the two structures over one another, but caution must be taken in order not to mix up the tokens.

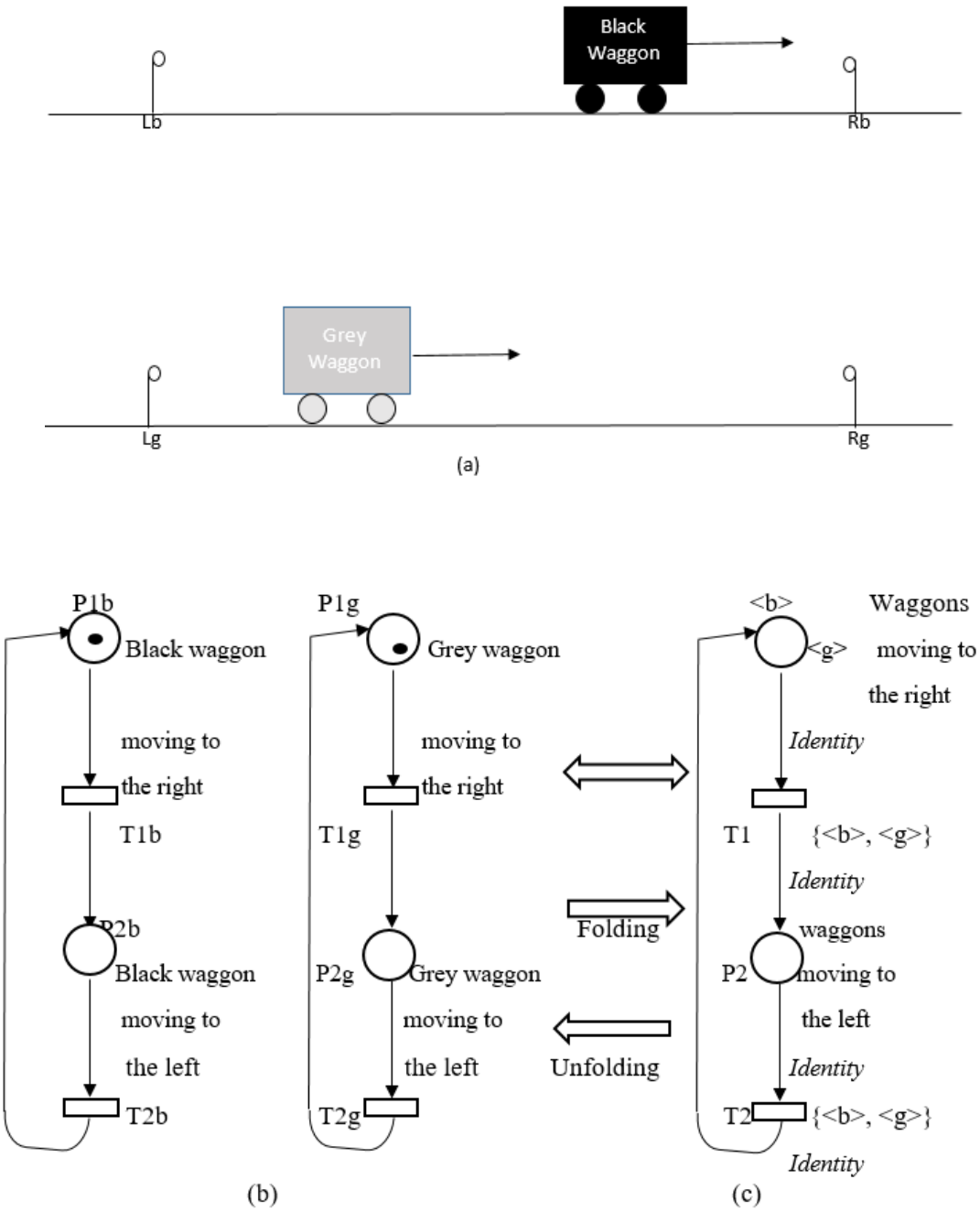

Figure 4: (a) The system. (b) Its ordinary Petri net. (c) Its colored Petri net

To distinguish the tokens, labels which are also referred to as colors are attached to them. As such, ‘b’ and ‘g’ are used to denote ‘black’ and ‘grey’ respectively. Generally, a color may be n-tuple, having complicated transformation functions [4].

2.2.2 Hierarchical Petri Nets

Creation of large and complex Petri net can be very complicated. Large CP-nets can be broken down into independent separate subnets, which gives a wider picture of the system. The subnets which are usually of a practicable size are kept on a different level arranged in hierarchies of layers.

In creation of hierarchical CP-nets, a small number of CP-nets can be used. Hierarchical Petri nets help with the management of large-scale models. Two types of design are usually employed when constructing a hierarchical Petri net: down-to-top and top-to-down design styles. Hierarchical CP-nets are multilayered nets.

2.2.3 Timed Petri Nets

Timed Petri nets, were first introduced by Merlin and Ferber [5]. It entails two-time values defined for each transition. When a Petri net transition fires in real time, the Petri net is referred to as a Timed Petri net. This implies that there is a random or deterministic firing-time linked with each transition. Here the tokens are removed from input places when fired and placed into output places when firing stops. Timed Petri nets take into consideration the distribution of tokens in places as well as in firing transition. Performance analysis of timed net is based on stationary probabilities of states. Timed Petri nets have continuously gained application in the modelling and verification of real-time concurrent systems.

2.2.4 Stochastic Petri Nets

Stochastic models are more preferable when system structure and initial conditions are less known and when system has less predictable macro-behavior. In dealing with biomolecular interactions, issues arises as to what degree the underlying biomolecular system is sensitive to changes in molecular density and randomness and randomness in occurrence of biomolecular reactions. Basically, stochastic Petri nets can be

considered as timed Petri net which has timing in stochastic values. The transition rate of stochastic transition is probabilistic in nature with no precise prediction. When taking into consideration uncertainty attached to data, stochastic Petri nets generates such random aspects which enabled transitions fire with exponentially distributed time delays [6]. The marking graph of a SPN is similar to a finite Markov Chain [7].

2.2.5 Continuous Petri Nets

A continuous Petri net can be formally defined as 5-tuple $R = (P, T, Pre, Post, M_0)$ where $P = \{P_1, \dots \dots, P_n\}$ is the set of finite and non-empty places, $T = \{t_1, \dots, t_m\}$ is a finite and non-empty set of transitions, $Pre: P \times T \to Q$ is weight function which assigns a positive rational number to arc from the input place to the corresponding transition, $Post: P \times T \to Q$ is the weight function which assigns positive rational number from transition to output place, $M_0: P \to R$ is the initial marking.

The continuous Petri net consists of continuous place and continuous transition. It was initially introduced to solve the state explosion problem, which occurs when a set of reachable markings increases exponentially to cause memory overflow. With time, it was realized that continuous Petri net is an efficient tool for biomolecular modelling [3]. This is to create a representation of a smooth change in concentration level of biological substances.

The number of markings assigned to C-places is positive real number values and not integer values, thereby making transitions to fire continuously. The fact that continuous change can be measured in terms of rational numbers makes it easy for quantitative modelling of biological systems.

Continuous Petri nets are suitable for modelling liquid flows, continuous production of a machine and other problems involving smooth change of the systems parameters. There is relationship between continuous Petri nets and ordinary differential equations; both methods can be used for modelling qualitative description of biochemical reaction network [5]. Figure 5(a) illustrates a Petri net model using only continuous places and transitions, 5(b) continuous transition and 5(c) continuous place.

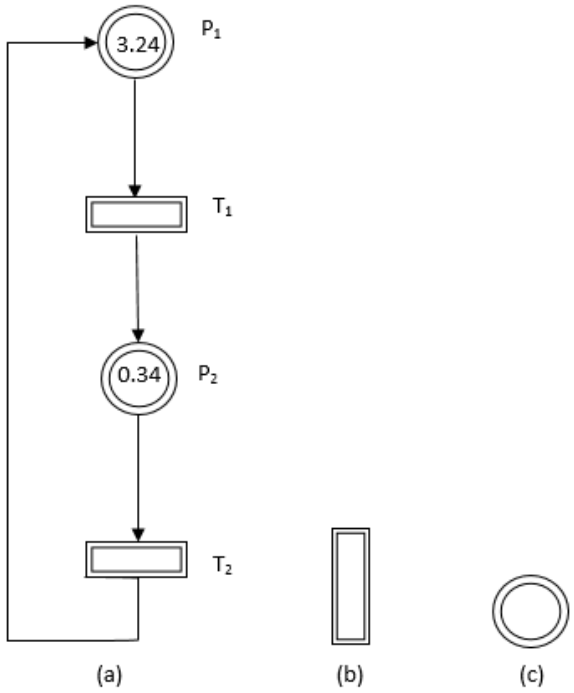

Figure 5: (a) Continuous Petri net (b) Continuous transition (c) Continuous place

2.2.6 Hybrid Petri Nets

An hybrid Petri net consists of both discrete and continuous Petri net fragments. This implies that the places and transition can be discrete or continuous. The discrete place consists of tokens while the continuous place is assigned real value numbers.

In modelling continuous flow system such as chemical reactions, biological processes, drug synthesis, water supply, gas condensing and oil production, continuous Petri net are very useful. Quite often, continuous-flow systems comprise different structured processes involving discrete as against continuous activities such as presence or absence of mutation or counter like mechanisms. These systems can be modelled in terms of hybrid Petri nets that incorporate continuous and discrete capabilities.

An hybrid Petri net is defined by$B = \{R, fnt, fat, m_0, \dots, f_c\}$

• R is a Petri net defined by $\{P, T, A, Pre, Post\}$ where

• P is a finite set of places with $|P| = nP < \infty$.

• T is a finite set of transitions with $|T| = mT < \infty$.

• $A \subseteq (P\ x\ T) \cup (T\ x\ P)$: finite set of directed arcs.

• $P \cap T = \emptyset\ and\ P \cup T \neq \emptyset$.

• $Pre: P\ x\ T \rightarrow R\ or\ N_+$ is a function that defines the weight of arcs from a place to a transition.

• $Post: P\ x\ T \rightarrow R\ or\ N_+$ is a function that defines the weight of arcs from a transition to a place.

•$m_0: P \rightarrow R\ or\ N$ is the initial marking.

•fnt: This function indicates the node type which can be either discrete or continuous.

•fat: This function defines the arc type.

A continuous net fragment can be affected by the discrete fragment and vice versa in a hybrid Petri net. When a D-transition t is affected by a C-place p, then $pre(t,p) = post(t,p)$ occurs. This is an indication that C-place can enable or disable D-transition, while firing of the D-transition cannot modify $m(p)$.

Condition for enabling a D-transition in an HPN is given by $m(p) >= Pre(p,t)$. This definition does not differentiate if p is a C-place or D-place. The enabling degree of a D-transition remains same to that of a discrete Petri net. An arc joining a C-place to a D-transition can be understood as being an enabling condition relating to a threshold that must overtake C-place marking. C-transition t is enabled only if each D-place p satisfies $m(p) >= Pre(p,t)$ and each C-place p satisfies $m(p) > 0$. The enabling degree of a C-transition in an HPN is said to be the same to that of a continuous Petri net. Conversion of a marking from discrete to continuous or vice versa is possible when a D-transition fire. Example of an hybrid Petri net is illustrated in figure 6 below:

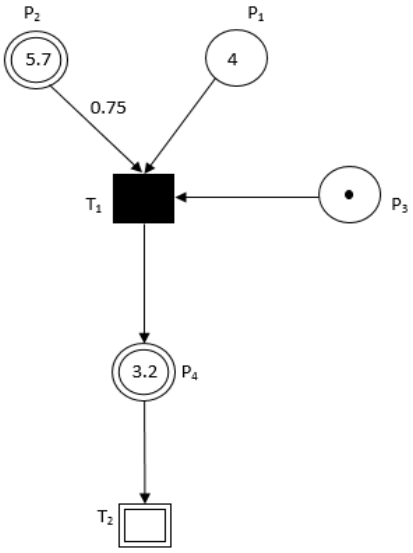

Figure 6: Example of an hybrid Petri net

2.3 Properties of Petri Nets

We distinguish between behavioral and structural properties. Behavioral property is the one which depends on the making or the initial state of the system. In contrary, structural property considers net structure, interconnection between its components and does not take into account system's states.

For instance, reachability is important when we want to know whether a marking can or cannot be reached from the initial marking. This enables one to know if there is a firing sequence that can lead the net to a desired marking [3].

On the other hand, boundedness is useful when we want to define the size of the system while safety is important to reveal some design errors. Notice that a Petri net is k-bounded with respect to an initial marking M_0, if the number of tokens in any of its places never exceeds k for any marking. A Petri net is safe if it is 1-bounded. Safety is a particular case of boundedness. If Petri net is safe, then none of its places receives more than one token.

Reversibility property is often demanded by many applications. If for each marking $M \epsilon R(M_0)$, M is reachable from M, a Petri net (N, M_0) is said to be reversible. In reversible net, one can always get back to the initial marking or state. In many applications, it is not important to get back to the initial state since one can get to some home state. Boundedness, liveness, and reversibility are independent of each other, a reversible net can be live or not live and bounded or not bounded.

Coverability is roughly related to 1-liveness. A marking M in a Petri net (N, M_o) is said to be coverable if there exist a marking M' ϵ R (M_0) such that M' (P) $\geq$ M (P) for all p ϵ P. Let M be the minimum marking to enable transition t ϵ T, then t is dead if and only if M is not coverable. This t is 1- live if only M is coverable.

A Petri net is persistent if for any pair of enable transitions, occurrence of one of them does not disable the other. Persistence is significant in analysis of parallel programs and asynchronous circuits. In this sense, persistency is somewhat similar to conflict-free nets.

Fairness: We have two basic fairness concepts, bounded fairness (B- fair) and unconditional fairness. Two transition t_1 and t_2 are said to be inbounded fairness relation when the maximum number of times that either one can fire while another is not is bounded. A firing sequence δ is said to be unconditionally fair if every transition in the net appears infinitely often in δ. The relationship between the two fairness types is that "every B-fair net is unconditionally fair net and every bounded unconditionally-fair net is a B-fair net" [8].

Structural Properties of Petri Nets.

These properties are of practical importance especially in design of manufacturing systems. Structural properties can be verified using algebraic techniques. Liveness, boundedness, conservativeness, repetitive, consistency, and controllability are among structural properties [7].

A Petri net is structurally live if there exists marking in which it is live. It worth to mention that live Petri net is definitely structurally live. In general, it is not so easy to check whether a Petri net is structurally live. If a transition is not live, then it is in a state of deadlock. The absence of deadlock is guaranteed by presence of liveness. This means that every transition of the Petri net can fire an infinite number of time. Any transition can be leave at one of the following levels. Examplle of a non-live and live Petri net is shown in figure 7 and figure 8 respectively.

Level 0: This means that *t* is a dead transition and it can never fire.

Level 1: It can fire once and there exist a marking $M \in R(M_0)$ so that *t* is enabled. A Petri net is said to be live at level *i* if all the transitions are live at level *i*, and level *i* refers to as level 1.

Level 2: there exists a firing sequence that consist of *t* at least *n* times where *n* is any positive integer.

Level 3: *t* can fire infinitely many times if there exist an infinite-length firing sequence but it can be blocked.

Level 4: There is no way it can be blocked and it can occur infinitely many times. It is said to be the strongest one.

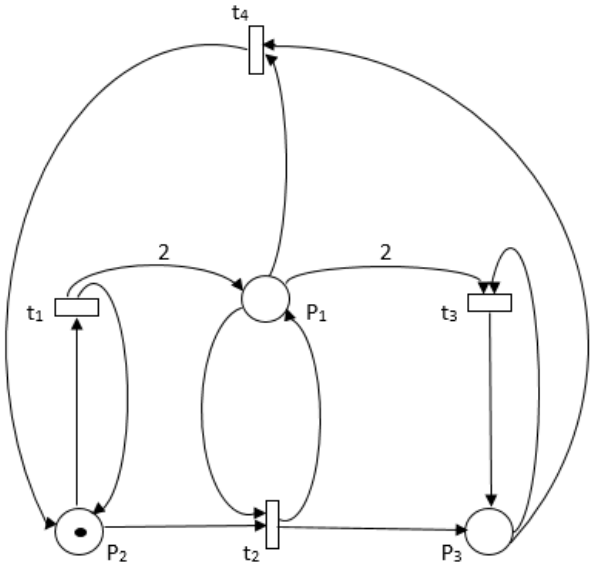

Figure 7: An example of a non-live Petri net

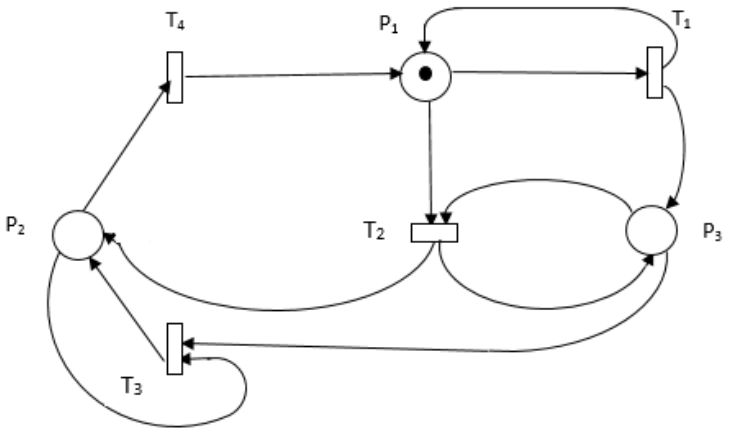

Figure 8: An example of a live Petri net

A Petri net is structurally bounded when it is bounded for any initial marking Mo. A Petri net that is structurally bounded is bounded but the reciprocal is not so. For a net to be structurally bounded, it must remain bounded for all possible initial marking, unlike structural liveness.

A Petri net is said to be conservative when all transitions fire tokens-preservingly, since the same number of tokens removed from the input place is the same as the number of tokens added to the output place. Conservativeness is closely related to structural boundedness property. A conservative Petri net is structurally bounded.

Repetitivity is a property that is closely related to structural liveness. A Petri net is repetitive if every transition occurs infinitely often in any firing sequence δ from Mo. Consistency occurs when there exists a Mo with a firing sequence δ from Mo to Mo such that every transition occurs at least once in δ. Controllability occurs when a Petri net is said to be totally controllable if any marking is reachable from any other marking.

2.4 Analysis Method

In Petri nets, methods of analysis could be classified into the following three; Coverability (Reachability), the matrix equation approach, and the reduction or decomposition techniques.

Coverability involves majorly the enumeration of all reachable markings for their coverable markings. This should be able to be applicable to all classes of nets, though limited to modest-sized net due to the complexity of the state-space explosion. State equation and reduction methods are useful enough, but both methods can be efficiently applied to restrict class of Petri nets.

2.4.1 Reachability and Coverability Methods

Consider a Petri net (N, M_0). From M_0 we can obtain as many "new" markings as the number of enable transitions. This procedure applied repeatedly results in a reachability tree composed of all those marking reachable from the initial marking. It is a herculean task to draw a reachability tree for an arbitrary Petri net with unbounded and live Petri nets. Since reachability tree can be drawn only for modest-size Petri nets with nice properties such as boundedness and liveness. An example of a reachability of a Petri net is given in Figure 9.

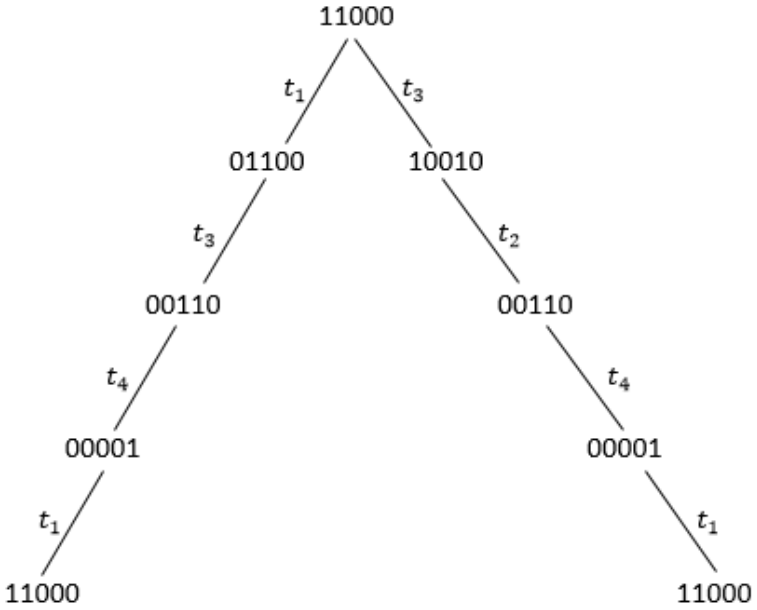

Figure 9: An example of a reachability tree of a Petri net

Most of the net properties can be decided using its reachability tree. The disadvantage of this approach is that it is rather exhaustive to draw reachability tree. Apart from this, it is not possible to draw reachability tree for unbounded nets [3]. Instead, coverability tree method can used. Coverability tree is obtained by folding reachability tree [8]. The price of this reduction is loss of some information so that some properties cannot be verified anymore. For example, Coverability tree cannot be used to check for liveness and reachability problems. In general, due to the information lost by the use of the symbol, the reachability and liveness problems cannot be solved by the coverability tree method alone, two different Petri net can have the same coverability tree and the same coverability graph.

2.4.2 Method of Matrix Invariants

In describing and analyzing completely the dynamic behavior of Petri nets by some equations, matrix equations that govern the ever-changing behavior of concurrent systems modelled by Petri net shall be examined. The solvability of these equations, though limited, is partly due to the non-deterministic nature found in Petri net models

and because of the limitations that solutions need be located as a non-negative integer. Generally, matrix equations are discussed in line with the fact that Petri net is pure or is made pure by including a replica pair of a transition and a place.

Given a Petri net with *n* transition and *m* places, an incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix which is given as $a_{ij} = a_{ij}^{+} \quad - a_{ij}^{-}$, where $a_{ij}^{+} = w(i,j)$ is the arc weight from transition i to j, and $a_{ij}^{-} = w(j,i)$ is the arc weight form transition j to i. It can be observed that a_{ij}^{+} , a_{ij}^{-} , a_{ij} is the number of tokens added, removed and changed in when transition *i* fires. The state equation for a Petri net is expressed as:

$$A^{T} \cdot x = \Delta M \quad M_d$$

Where $\Delta M = \ M_d - M_0$ which is the number of tokens between the two states M_d and M_0. M_0 is the initial marking and M_d is the destination marking. A^T is the transposed matrix. x is a $n \times 1$ column vector which comprises of non-negative integers also known as the firing vector. Generally, M_d is reachable from M_0 when there exist a vector *x* such that a non-negative solution is obtained from the related matrix. For general Petri nets, a non-negative solution does not necessarily give a desired reachability from the initial marking M_0 [9].

2.4.3 Reduction and Decomposition Method

One of the main limitations of Petri net is the complexity problem, in addressing this, it became necessary to create changes or transformations method that allows a sort of stepwise reductions and preservation of the system properties. This process decreases the reachable state space of the net, making analysis easier as a result of the simplified net.

A reduction is brought about when there is the need to facilitate the analysis of a large system. The reduction of a Petri net can be achieved without changing the original properties. Part of the basic properties upon which Petri nets are defined are liveness, boundedness and proper termination. When analyzing them, modelling errors are detected, thus revealing many desirable properties of the systems modelled by a Petri net. The analysis of a large Petri net is often complex, thus, the basic reason for Petri net reduction is to reduce the complexity of analysis, since a reduction helps to preserve the properties of the net.

Chapter 3

QUANTITATIVE MODELLING WITH PETRI NETS

Quantitative analysis can be defined as the measurement of the quantities of particular constituents present in a substance [10]. Quantitative methods emphasize objective measurements and the statistical, mathematical, or numerical analysis of collected data or by manipulating pre-existing statistical data using computational techniques. It is an important aspect of analysis of any product, substance, a chemical or a drug formulation. Analysis (Scientific) is a qualitative and quantitative estimation of any compound or substance by a defined and accepted procedures under standard set of conditions.

Quantitative modelling is a critical factor and plays a very important role in interpreting experimental data. It is based on continuous concentration and reaction rates. In converting Petri nets to quantitative models, tokens will not be used but replaced with real numbers value and represent concentration. Petri net can be efficiently used for quantitative modelling of biochemical processes. In this context, the firing rule is no longer a simple discrete event, but contains functions to describe complex of chemical reactions [11].

3.1 Law of Mass Action

Chemical reaction describes the conversion of chemical substances which are also known as reactants, educts or substrates in case of catalyzed reactions. The substances created during this chemical reaction are referred to as products [10].

Spontaneous reactions are known as exothermic, otherwise reactions are endothermic. There are two fundamental types of chemical reactions according to the amount of participating reactants, which are; Unimolecular (1st order) reactions and bimolecular (2nd Order) reactions.

According to the Law of Mass Action "speed of chemical reaction depends on concentration of reactants presented in reactions or better say the rate of conversion of masses of generic chemical reactions is proportional to the product of the masses of the reacting substances." [11]. The concentration of reactants affects the reaction speed: speed increases with increase of concentration. Generic chemical reaction has the following view:

$$mA + nB \xrightarrow{k} C$$

where A, B and C are reactants, m and n are stoichiometry coefficients and k is the reaction constant. Rate of chemical reaction can be represented as follows:

$$r = k[A]^m[B]^n$$

where r is the rate, speed or velocity of reaction, $[A]$ is the concentration of substance A and k is the reaction constant. It must be also noticed that rate of a chemical reaction is measured in terms of concentration of its reactants [12].

Example for law of mass action is given below:

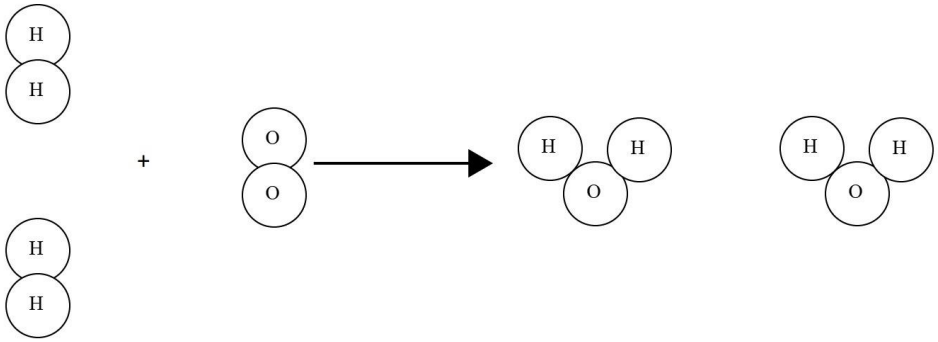

Figure 10: Illustration of the formation of water molecules from hydrogen and oxygen atoms

4 hydogen atoms +2 Oxygen atoms = 4 Hydrogen atoms + 2 Oxygen atoms

Four hydrogen and two hydrogen atoms gives four hydrogen atoms and two oxygen atoms.

The following reaction types can be distinguished:

3.1.1 Zero Order Reactions

Zero order reaction is the simplest chemical reaction where the reaction rate does not depend on the concentration of reactant.

$$NULL \longrightarrow B$$

3.1.2 First Order or Unimolecular Reactions

First order reaction represents conversion of an unstable molecule of type A to molecule of type B. The conversion may represent various complex processes, for example, chemical modifications or structure changes, whose details are unknown or the reaction may be part of a bigger reaction that produces molecules of type A which are unstable. The reaction rate k is proportional to the concentration $[A]$ of the reactant [11].

$$A \xrightarrow{k} B$$

The ordinary differential equation giving concentrations of both A and B can be written as:

$$\frac{d[A]}{dt} = -\frac{d[B]}{dt} = -k[A]$$

where $[X]$ represents the concentration of substance X. The Petri net representation of a unimolecular reaction is shown in figure 11:

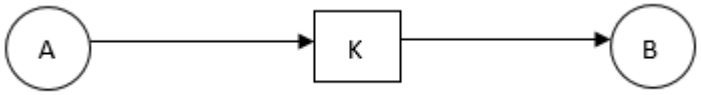

Figure 11: Petri net model of a unimolecular reaction

3.1.3 Biomolecular or Second Order Reaction

This is a type of reaction between two entities of the same type or different type in a reaction system.

$$A + B \xrightarrow{k} C$$

In this case the reaction rate is proportional to the concentrations of its reactants. The ordinary differential equation is as follows:

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = -k[A][B]$$

$$\frac{d[C]}{dt} = k[A][B]$$

where k is the kinetic rate constant coefficient

Examples of bimolecular reactions include: explosion, bifurcation, waves, oscillation, etc. The Petri net representation of a biomolecular reaction is shown in figure 12.

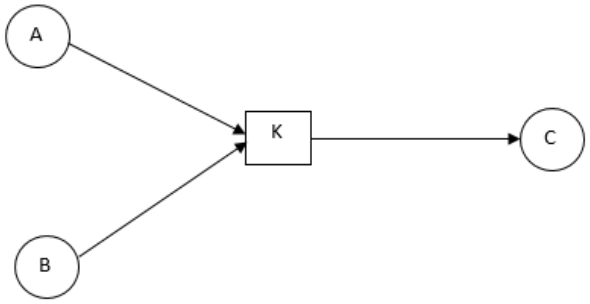

Figure 12: Petri net model of a bimolecular reaction

3.1.4 Reversible Unimolecular Reaction

$$A \underset{k_2}{\overset{k_1}{\rightleftharpoons}} B$$

Given reversible reaction represents two reactions:

$$A \xrightarrow{K_1} B$$

$$B \xrightarrow{K_2} A$$

where A and B are reactants, K_1 and K_2 are reaction rates constants. Petri net model of a reversible unimolecular reaction is given in figure 13:

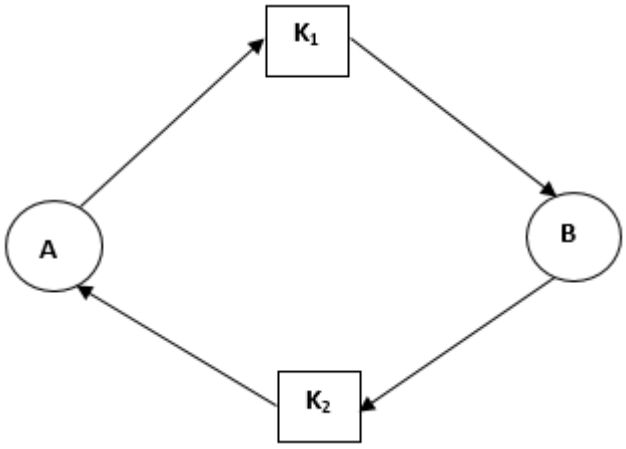

Figure 13: Petri net model of a reversible unimolecular reaction

Ordinary differential equation representing this process looks like:

$$\frac{d[A]}{dt} = -\frac{d[B]}{dt} = -K_1[A] + K_2[B]$$

3.1.5 Reversible Bimolecular Reaction

Biomolecular Reaction can also be reversible just like molecular reaction.

$$A + B \underset{k_2}{\overset{k_1}{\rightleftharpoons}} \mathrm{C}$$

The ordinary differential equations representing reversible molecular reaction is as follows:

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = -K_1[A][B] + K_2[C]$$

$$-K_1[A][B] + K_2[C] = K_1[A][B] - K_2[C]$$

The Petri net model of a reversible bimolecular is shown in figure 14:

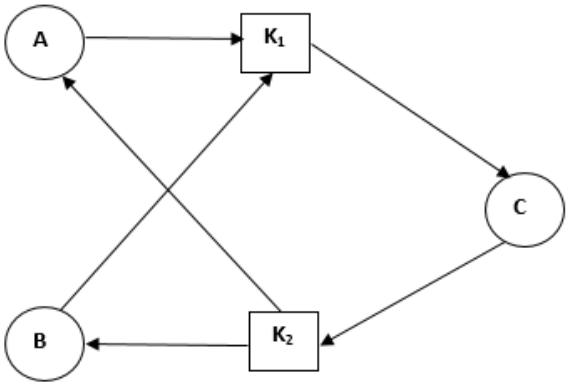

Figure 14: Petri net model of a reversible bimolecular reaction

Starting with initial concentrations of species A, B, and C the reaction will continue till equilibrium, which is called ***steady state*** is reached [11].

3.2 Steady State

At steady state, the derivative of the concentration with respect to time is zero, meaning neither concentration of the input and output components remain unchanged [13]. In simple systems the steady state is approached by state variables gradually decreasing or increasing until they reach their steady state value. In more complex systems state variable might fluctuate around the theoretical steady state either forever (a limit cycle) or gradually coming closer and closer [14]. It theoretically takes an infinite time to reach steady state, just as it takes an infinite time to reach chemical equilibrium. Reaction has to occur for steady state to develop.

$$\frac{d[A]}{dt} = \frac{d[B]}{dt} = \frac{d[C]}{dt} = 0$$

Steady states are the solutions of the algebraic equation:

$$\overrightarrow{f}(\vec{x}) = 0$$

It also defines values of the concentration at which their time deviation becomes zero. The illustration of steady state and transiet state is shown in figure 16.

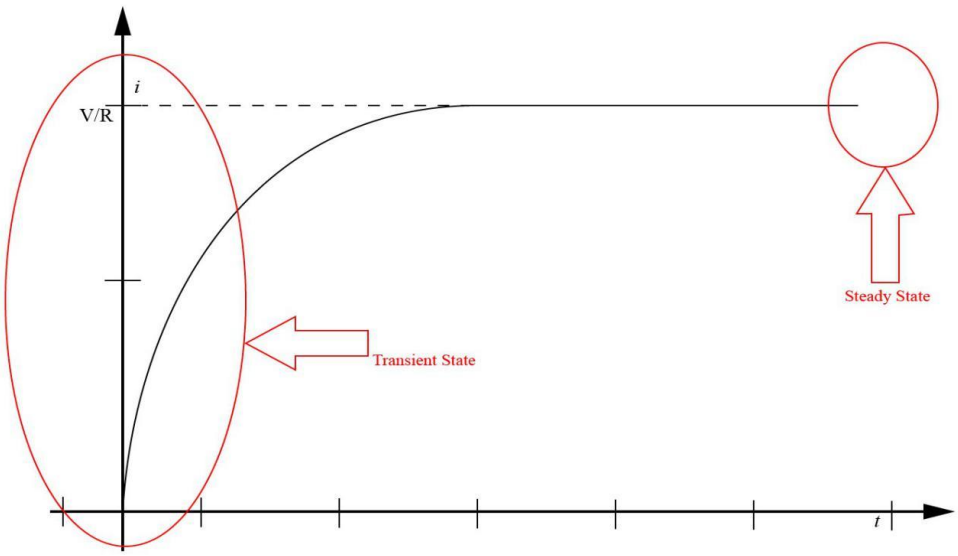

Figure 15: Steady state and transient state

The transient state is the brief initial state during which the amount of reactant concentration is changing fast with time. The steady state is when the concentration remains the same [10].

3.3 Dynamic Equilibrium

Chemical Equilibrium is a dynamic equilibrium. This occurs when two different functions have the same value. In a system at chemical equilibrium, the net reaction rate is zero (products transform into reactants at the same rate as reactants transform into products), while no such limitation exists in the steady state concept [15].

The concept of chemical equilibrium was developed after Berthollet (1803) found that some chemical reactions are reversible. For any reaction mixture to exist at equilibrium, the rates of the forward and backward (reverse) reactions are equal.

This is shown in the following chemical equation with arrows pointing both ways to indicate equilibrium [12].

$$\alpha A + \beta B \rightleftharpoons \sigma S + \tau T$$

A and B are reactant chemical species, S and T are product species, and α, β, σ, and τ are the stoichiometric coefficients of the respective reactants and products.

Rate of forward reaction= rate of reverse reaction.

Net reaction rate = 0

Equilibrium Point: This is when neither the concentration of input or output of component changes. Example of a dynamic equilibrum reaction is shown in figure 16.

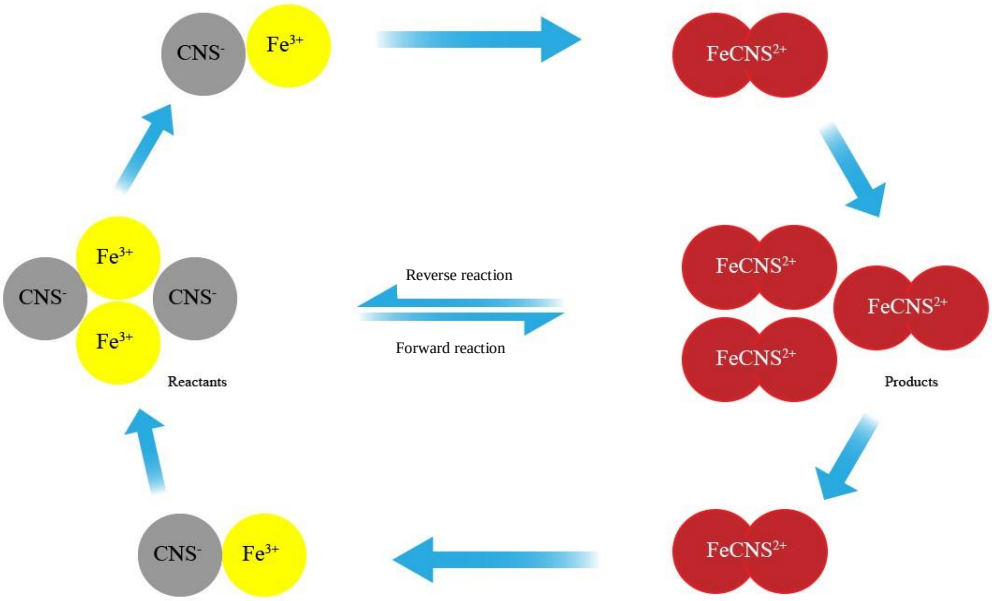

Figure 16: Dynamic equilibrium showing the forward and reverse reaction [12]

3.4 Validation

Validation is the choosing of the model parameter which result in the right parameter such that the concentration agrees with those obtained in biological experiment. It is the action of checking or proving the validity or accuracy of something. Figure 17 shows the various types of validation techniques.

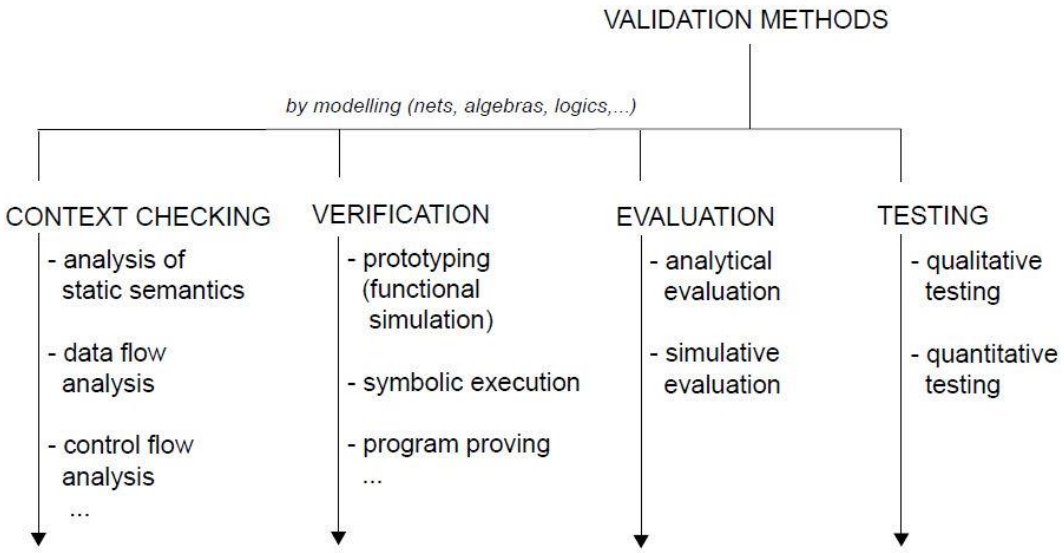

Figure 17: Validation techniques

3.5 Review of Existing Software

There are various software available for Petri net simulation. Many tools use Petri nets as a base for simulation as a result of the ease in which Petri nets can model the characteristics of a system and how well its model lends itself to discrete event simulation.

3.5.1 HPsim

HPsim is a simple yet robust software tool used for Petri net simulation by beginners. It consist of a graphical editor (combined with basic editing and simulation features), token game animation, fast simulation and simple performance analysis. The graphic editor supports both print and zoom function. Graphic objects can also be deleted, moved and repositioned. It supports Place/transition with limited capacity for place [16]. There are edges with different weights and timed transitions. HPsim supports both timed and stochastic Petri nets**.**

In HPSim, "firing delays" can be associated with transitions (property of the transition object). When the time specified has passed from the moment the enabled transition began to take place, the timed transition fires. We have a timed network if the specified time is constant; the timed network is referred to as stochastic if the time specified is in accordance with a given distribution. It is essential that the time count for a transition is from the moment the transition is enabled, regardless of when a certain mark “entered” the position before the transition [16].

One of the advantages of using HPsim is that the graphical interface helps in error detection and model development. Simulation result can be saved in Microsoft excel which aids in the analysis process.

3.5.2 SimHPN

SimHPN is a very user friendly Graphical User Interface (GUI) with almost all the procedure related to structural analysis and simulation accessed through the use of the GUI controls. It consist of a Graphical Editor, State spaces, Place and transition Invariants Structural and Simple Performance Analysis, and Fast simulation. It is a selection of tools dedicated to analysis and simulation, discrete process created by hybrid and continuous Petri net. SimHPN is embedded in MATLAB which makes it easy to create statistical, algebraic and graphical instrument which can be found in the MATLAB environment. The data obtained after simulation can be exported to the MATLAB workspace for subsequent analysis. SimHPN also provides some analysis algorithms apart from simulation. The algorithms include update clocks, Simulation of HPN using a variable sample time, Computation of minimal P-semiflows, and Reduction of places from threads [17].

The main features include the various types of visualization options, Optimal Sensor placement and steady-state, Import functions from various types of graphical Petri net editors, Computation of throughput bounds and P-T-semiflows, use of Infinite server (discrete transition), finite server and product semantics when simulating continuous Petri nets. For discrete transition, deterministic delay with single server semantics can also be used. The model description data are all MATLAB variables. Simulation of a hybrid Petri net can be done using SimHPN without opening the GUI, this is done from the MATLAB prompt. This is usually done when implementing a script and simulating at the same time depending on the input model.

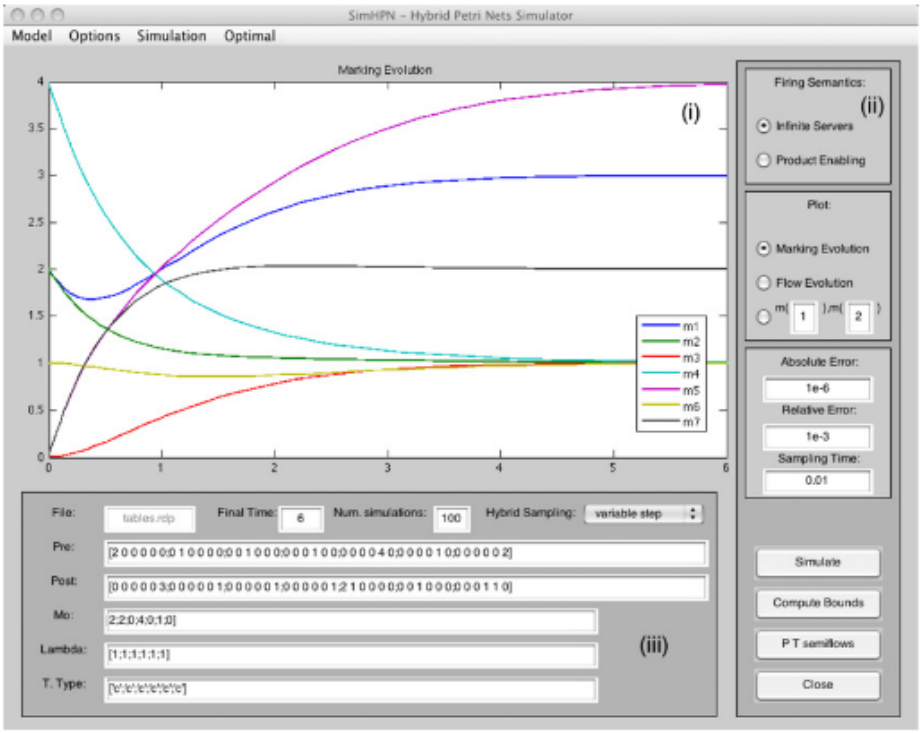

Figure 18: Sketch of the main window of SimHPN

Figure 18 shows the sketch of the main window of SimHPN. It consist of the menu bar and three control panels which are drawing area, options panel and the model management panel. There are 4 drop menu bars at the top of the windows including model, options, simulation and optimal. A SimHPN supports plain continuous, discrete and hybrid Petri nets.

3.5.3 HiQPN

HiQPN is also known as hierarchically combined queuing Petri net. The HQPN formalism was developed by Dr. Falko Bause [18]. It combines both queuing network modeling formalism and Petri net and can be used for analyzing complex structures [19]. This helps in avoiding difficulties such as description of scheduling strategies using Petri net elements. Due to its hierarchical specification, it support both qualitative and quantitative (performance) analysis and hierarchical model specification. If the QPN has a hierarchical structure, the difficulty of a quantitative

analysis (which is based on numerical Markov chain analysis) is remarkably reduced [18].

The main idea of the analysis is to replace large matrix generated the Markov Chain by smaller matrices, with the smaller matrices representing a sub model which are combined together. This leads to an extension of the model formalism towards Hierarchically Combined Queuing Petri Nets (HQPNs). It support both High-level and Stochastic Petri nets. HiQPN consists of Graphical editor, Token game animation, Transition and place invariants, Interchange file format, Advanced Performance analysis, and state spaces. QPNs are superset of Generalized Stochastic Petri Nets (GSPNs) also combining queues into places. HQPN is the new version of QPN-Tool which allows the processing of places by QPN queues or subnets. A very important characteristics of the hierarchical technique is that it leads to precise results which does not depend on model consistency. HQPN are obtained from the general formalism of QPN. Example of a QPN is shown in figure 19.

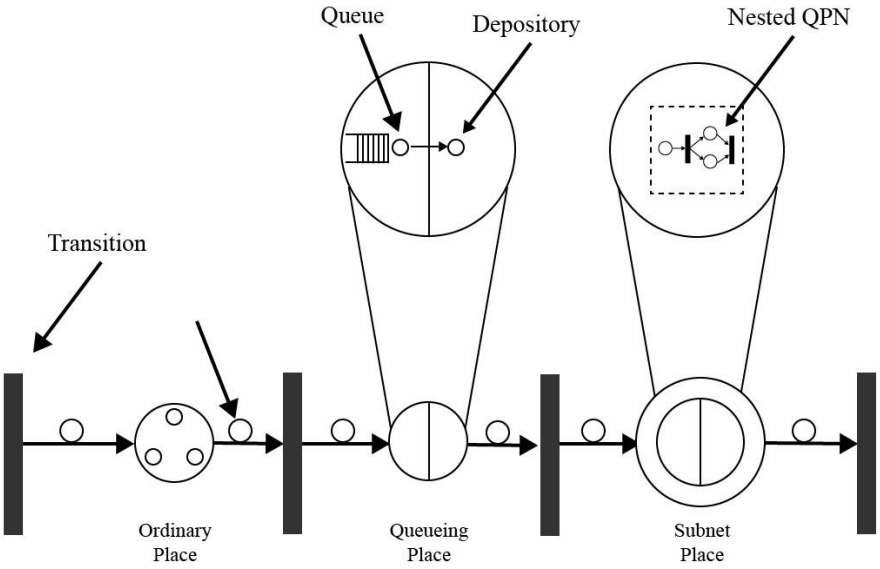

Figure 19: Example of a QPN

$$QPN = QN + PN$$

Where QN is Queuing networks

PN is Petri Net

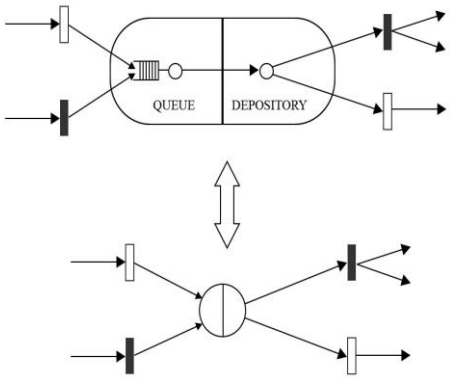

Figure 20: Queuing place of a QPN and its shorthand notation

A processing device is modelled as a queuing place. This is illustrated in figure 20.

Queuing Place = Queue + Depository.

Conditions and event can be represented by transition that can move a request token to the next queuing place.

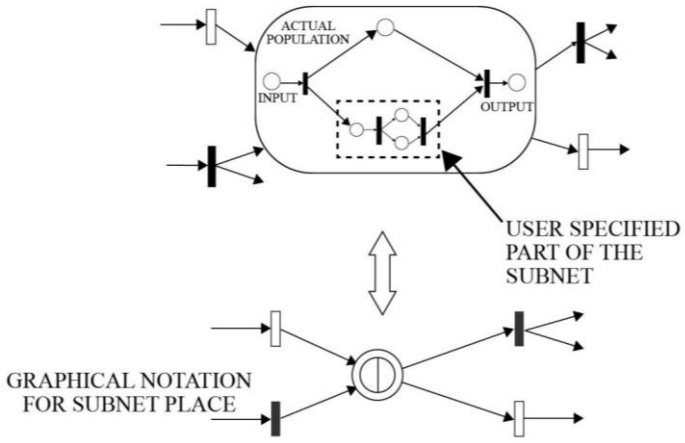

Figure 21: Subnet place of a HQPN and its shorthand representation

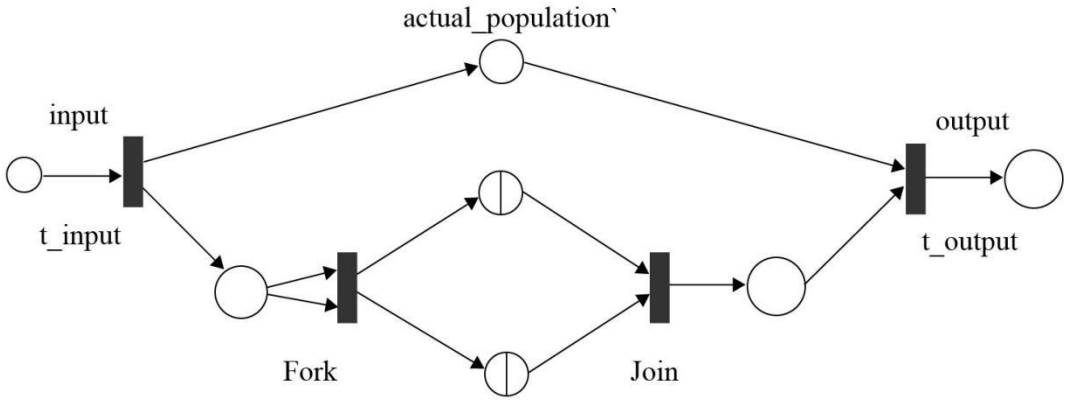

Figure 22: Isolated HQPN subnet

A subnet place is a timed place that consist of an entire HQPN subnet instead of one queue. The subnet place of the HQPN subnet has an output place and an input place that are ordinary places of a colored Petri net (CPN). This is given in figure 21. After the firing of an enabled transition, tokens that are been added into the subnet place is then transferred to the input place of the HQPN. The tokens present in the output place can only be used by the output subnet of the transition. This implies that tokens present in other part of the HQPN subnet cannot be used in the output transition of the subnet place. Example of an isolated HQPN subnet is given in figure 22. The incorporation of the subnet into the real QPN formalism leads to the hierarchical structure of the QPN framework/prototype. There are several levels of hierarchical structure but two levels are usually referred to for the sake of simplicity: The high level is known as High Level QPN (HLQPN) and the HQPN subnet of the subnet places is referred to as Lower Level QPN (LLQPN) [20]. HQPN Supports analytical solution techniques and runs on Sun-OS 5.5.x / Solaris 2.

3.5.4 PetriSim

PetriSim is also known as My Pet reasonable interactive simulator. This models can be generated by the addition of a small fragment of Pascal code. Moby Turbo or better

still Borland Pascal 7 is needed to do this and only a PC version is available. PetriSim was created to support education of Object Oriented Programming, Discrete Simulation, and Petri nets [21].

PetriSim 1 was first created as a simulator and graphical editor of Petri Nets (the Place/Transition Nets). It was a very user friendly and enables intuitiveness which helps in creation, simulation, editing of any number of Petri Nets at a time. Although time control was the user's responsibility, it was feasible to add Pascal code snippets to build user framework. Most operations were performed by drawing on the screen using the mouse [22].

The Time Nets was introduced in PetriSim 2. The firing delay concept was used here by creating a certain delay between tokens that are been removed from the input places and adding the tokens to the output place of a transition that is been fired. The delay is usually created when the firing begins by the activated user code snippet. So it is possible to create a synchronizing and complex timing structure that are distinctive for discrete simulation prototype. User codes are used to perform all operation on user data and also to generate delays. The sequencing and timing of all the processes are controlled by the Time Net. It is advisable that the user have an idea about intermediate Pascal Programming and basic ideas of PNs [22].

Two major advancement were added with the advent of PetriSim 3 which are Openness of inheritance structure and Visual Programming. Visual programming gives a user friendly relationship between code snippets and transitions. A menu is open by just clicking the transition that gives two options to either start a text editor or edit/enter the ending and starting of the code snippets.

With PetriSim 3.1, the inhibitor arcs was generalized and more arcs were added together with removal of some bugs. The arcs are assigned weight which makes it possible to test if place has less than a fixed amount of tokens. This implies that it is feasible to test if a place has at least a fixed amount of tokens with the use of the testing arc [22].

The PetriSim 4 is almost the same as the previous version but with more added functionality which helps to ease the creation of user models. There is evaluation of statistics and transparent collection on the amount of tokens in a place, the lifetime of a token in a place, and firing life span of a transition. The user statistics of the model made of the place and transition is automatically made available if a transition is used as server and place used as queue. User programming is reduced and the only programming usually been done is writing the code snippets.

For most models, the need to write user code is removed with the advent of PetriSim 5 which is the latest PetriSim tool. During screen editing mode, parameters of Timed Nets like branching, probabilities, and random delays of firing etc. are input directly. Although state dependent behavior has to be programmed, otherwise most simulation models do not need any Pascal programming. Discrete simulation models can be created using PetriSim 5, this is usually done by creating a Timed Net on the screen. Intermediate Pascal programming is usually required together with basic knowledge of Petri Nets [22]. Figure 23 shows the main menu GUI of a PetriSim environment.

Figure 23: Main menu of PetriSim environment

3.5.5 HISIm

HISIm supports hybrid, Place/Transition Nets, High Level Petri Nets, and Petri Nets with Time. It works with the Java environment and it's readily available online for download. Its components include fast simulation, token game animation and graphical editor. It allows user to simulate and create with the use of GUI, hybrid Petri nets (HPNs). It integrates test arcs and inhibitor arcs and follows Demongodin and Koussoulas' definition of Differential Petri Nets (DPN). DPN integrate elements which helps with accuracy issues affecting other formalism and it gives more flexibility than other Petri net formalism [21].

The marking of the continuous place (also, known as differential places in DPN) can take both positive and negative real number, and the maximum firing speed of continuous transitions (differential transitions) can be nonlinear or linear function of the continuous place marking associated with it.

Inhibitory arcs and test arcs are introduced and used to enabled transition but not to their firing which means that it cannot be used to connect a transition to an output place but input place to transition. Inhibitor arc as the name implies prevent the firing of a transition if the marking of the input place is at least equal to the arc weight and also tokens are not consumed either. It can also be observed that connection between a discrete place and continuous transition is not possible. Test arcs don't generate conflicts and does not reserve tokens either which makes it possible for a transition to be disabled as a result of change in the condition on a test/inhibitory arc during its delay time [23]. It was developed by Spanish scientist and funded by the Education and science Ministry of Spain under the ICSI fellowship.

3.5.6 Snoopy

Tool Description: Snoopy is a simulation software which can be used to design and execute (simulate, animate) hierarchical graph-based system descriptions and Petri nets. Due to its generic design, it comes along with several pre-fabricated graph classes, especially some kind of Petri nets and other related graphs, and also facilitates a comfortable integration of further graph classes. Different types of graph classes can be used at the same time and different types of classes can be changed into another, this is to support object-oriented Engineering. Snoopy provides some features (hierarchical nodes, logical nodes), that are really useful for big and complex models, or models that has higher connectivity degree [24].

There are different types of Petri net classes that are ready for use and available, some of which are purely qualitative place/transition nets in line with the standard definition and a version augmented by four special arcs and three quantitative extensions which are continuous Petri nets, stochastic Petri nets, and time Petri nets. These classes enjoys exclusive animation or simulation features.

Snoopy exports to different types of analysis tools, some of which are: McKit, PEP DSSZ-CTL and DSSZ-LTL, LoLA, Maria, PROD, Charlie TINA, SBML/Level 2/Version 3, INA, and Microsoft Excel. Snoopy provides import from these tools or languages TINA, PED, SBML/Level 2/Version 3, APNN. It has a Graphical Editor, Token Game Animation and the simulation process is very fast. It can work in Linux, MS Windows XP and 2000, Macintosh, and Mac OS X environment [21]. It is available online for free download from [2].

Snoopy is a framework that can be used to construct and simulate discrete, continuous, hybrid, Petri net with time, Place/Transition Net, and stochastic Petri nets.The simulation results can be shown as table, *xy* plot or histogram

Discrete place may enable or disable a continuous transition but firing action does not change its marking. There are six different arcs available for use, altough they are referred to as edges. They are: Edge (Standard or normal edge), Inhibitor Edge, Test Edge, Modifier edge, Equal Edge, and Reset Edge. Arcs weight can be a real number different from integer.

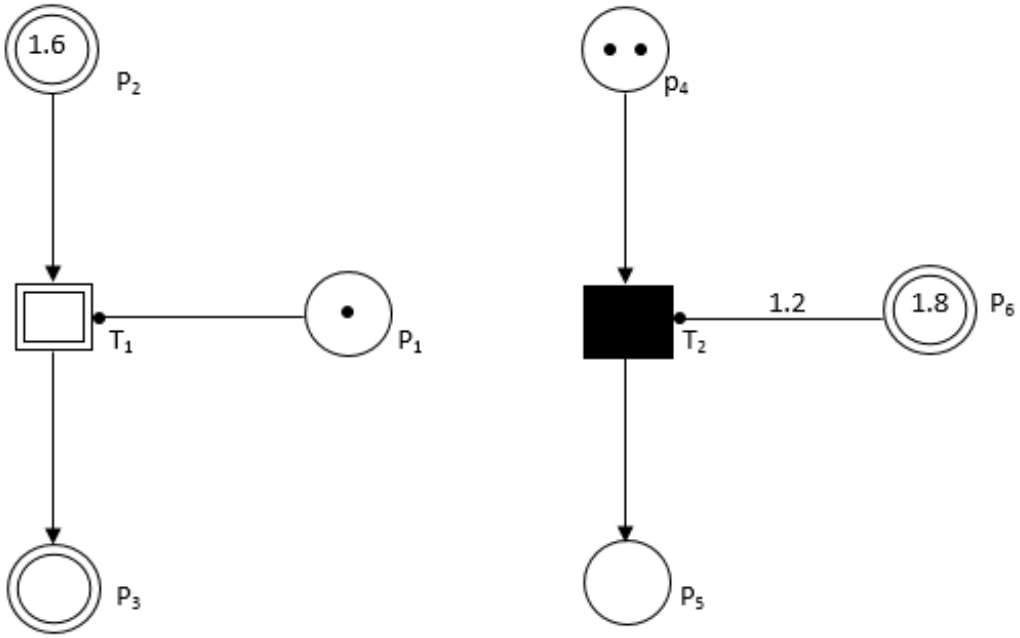

Figure 24: (a) Influence of discrete place on a continuous transition (b) Influence of continuous place on discrete transition

For change in a marking to be possible, the arc weight from a discrete place should be less than the quantity in the respective input place. Conversion of a marking of a discrete place to a continuous place is not possible with the use of continuous transition, but only with discrete transition. This is why regular arc cannot connect discrete place and continuous transition. It is generally believed that conversion of marking takes place, but change in marking did not actually happened. This is because the initial marking consists of mixture of both integer and real number and the subsequent marking consist of this mixture too. Since tokens was removed from discrete place but quantity was added to the continuous place, we can say that although marking is changed but the type of marking is not changed and conversion of marking did not necessarily take place.

Enabling Degree of Discrete and Continuous Transition

For a discrete transition, there exist a transition T_j with a marking m discrete transition:

The enabling degree for transition Tj for marking m, is denoted by q such that:

$$q \leq \min_{i:P_i \in {}^0T_j} \left(\frac{m(P_i)}{Pre\left(P_i, T_j\right)} \right) < q + 1$$

- $m(P_i) \geq Pre\ (P_i, T_j)$ (number of markings in the pre place should be greater than or equal to the product of arc weight and marks).
- If $q > 0$, then transition T_j is q-enabled.
- q is an integer [25].

For continuous transition, the enabling degree of transition Tj for marking, denoted by q is the real number such that:

$$q = \min_{i:P_i \in {}^0T_j} \left(\frac{m(P_i)}{Pre\left(P_i, T_j\right)} \right)$$

- $m(P_i) > 0$
- q is a real number
- if $q > 0$, then transition Tj is enabled: It is said to be q-enabled [25].

For a discrete transition having continuous and discrete preplaces to be enabled, there should be enough tokens or quantitiy in the preplaces which equals or is greater than the value of related arc weights. When discrete transition fires, the exact amount of arc weight value is deposited in the continuous output place. The firing of a discrete transition removes from the continuous place a quantity that is equal to the arc weight, while for a discrete place, the token is removed according to the ceiling function of the arc weight.

The firing of a continuous transition removes from the continuous place a quantity that is equal to the product of the arc weight and transition rate. For a continuous transition to be enabled there has to be enough quantity in each of the continuous input places, which is greater than the product of the arc weight and the transition rate.

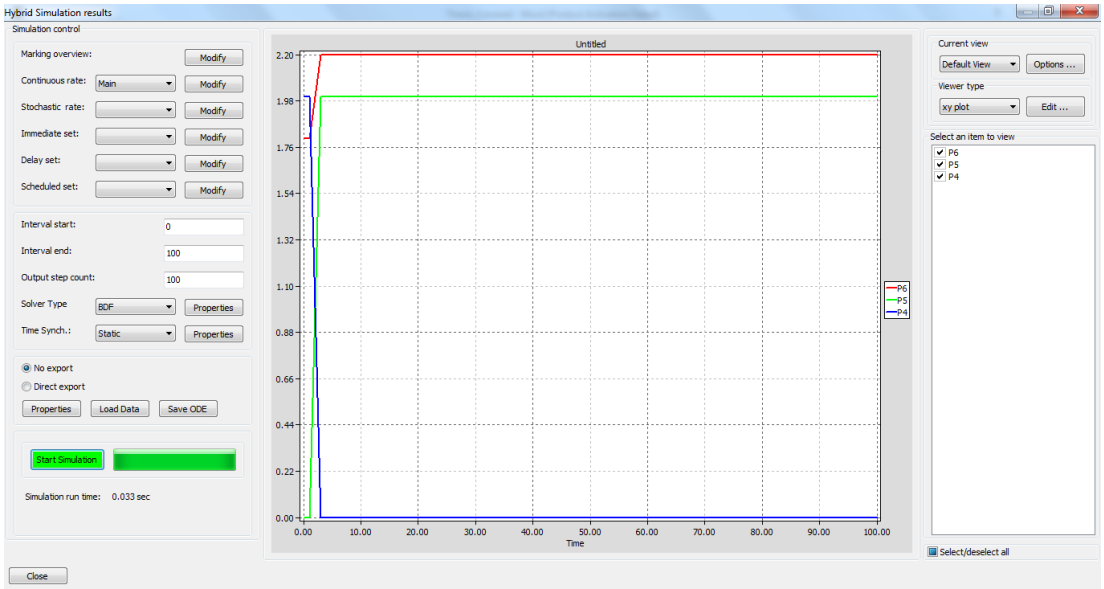

Figure 25: Snoopy's GUI (Graphical User Interface)

Figure 25 shows snoopy's graphical user interface. The simulation window is in two sections: Viewer sub windows and Configuration window. The left part consist of the configuration window and it allows the user to control and configure the simulation process. The viewer window which is on the right displays simulation results, which can be exported into CSV file and viewed in Microsoft Excel for further analysis of the results. The model editor allows creation of Petri net models where reactions are being replaced by transitions. It also provides features for configuration of the Petri net model with their initial state.

The simulation dialog has four parts which are: the simulator configuration, Import/Export, Model configuration and simulation state. After the creation of the model, the simulation dialog can be accessed from the menu bar. It is used for management and running of the simulation process.

The mass action function is set as default if no rate is assigned to a continuous transition.

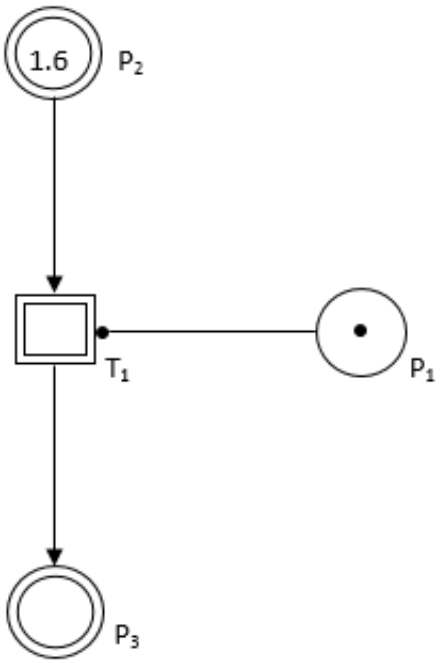

Figure 26: A Petri net with transition rate set as default (mass action)

A Petri net model was created with the transition rate set as default (mass action). This is illustrated in figure 26. The simulation result is shown in Table 1:

Table 1: Simulation result of a transition rate with mass action

Time	P1	P2	P3	K
Initial Marking	1	1.6	0	0.632136
1st Step	1	0.588583	1.0114168	0.632133
2nd Step	1	0.21652	1.38348	0.632142
3rd Step	1	0.079649	1.520351	0.632105
4th Step	1	0.029302	1.570698	0.632107
5th Step	1	0.01078	1.58922	0.632107
6th Step	1	0.003966	1.596034	0.632106
7th Step	1	0.001459	1.598541	0.632106
8th Step	1	0.000537	1.599463	0.632107
9th Step	1	0.000197	1.599803	0.632359
10th Step	1	7.26E-05	1.599927	0.632231
11th Step	1	2.67E-05	1.599973	0.631835
12th Step	1	9.83E-06	1.59999	0.63174
13th Step	1	3.62E-06	1.599996	0.632597
14th Step	1	1.33E-06	1.599999	0.631579
15th Step	1	4.90E-07	1.6	0.632653

The rate of reaction of mass action in snoopy is 0.63. It is calculated as thus:

$$K = \frac{Initial\ marking - Step\ 1}{Initial\ Marking}$$

$$K = 0.63 = \frac{1.6 - 0.588583}{1.6}$$

where K is the rate of reaction.

From this chapter, one can say that HPsim is suitable for beginners without any previous knowledge of Petri net, although it has limited place and a very user friendly Graphical user interface. With SimHPN, It is advisable to have a background knowledge of MATLAB since its Graphical User Interface is embedded in this environment. HiQPN is suitable for modelling large and complex models whichis usually broken down into subnets for easy comprehension. PetriSim was introduced in order to encourage the use of object oriented programming, thus having an intermediate knowledge of Pascal Programming is required for PetriSim users.

HISim helps to solve accuracy issues affecting other formalism mainly because its places can take on both positive and negative real number values.

Snoopy software was choosen because of its ability to work with discrete, continuous and hybrid Petri net. Although connection between discrete place and continuous transition is not possible with use of standard arc, so read arc is used instead which acts like a loop.

Chapter 4

CASE STUDIES

In this chapter, we give details about the three case studies used in the course of this thesis. The case studies were introduced in order to compare the interrelationship between discrete and continuous components. We discuss the modelling and simulation of the case studies using Snoopy software as a modelling and simulating tool after which the result is exported using CSV to Microsoft Excel. These case studies are: The bottling system, Production system and Olive soap production system.

General System Requirement

Windows PC

- 64-bit Win 10 OS
- Intel core i7 CPU running at 2.50 GHz
- 8 GB RAM

4.1 Bottling System

Figure 27 illustrates the Petri net model of the bottling system. The barrel which is P_4 contains 57 litters of wine and it is been put inside bottles with 0.75 liters capacity each. The 0.75 is represented by the arc weight from P_4 to T_2. T_2 is only enabled if there is at least 0.75 marks in P_4 and a token in P_1. In the initial marking, there is no token in P_2, which means there are no full bottles yet. This only occurs when the transition is enabled and firing takes place. P_3 has one token which enables or disable the transition T_2.

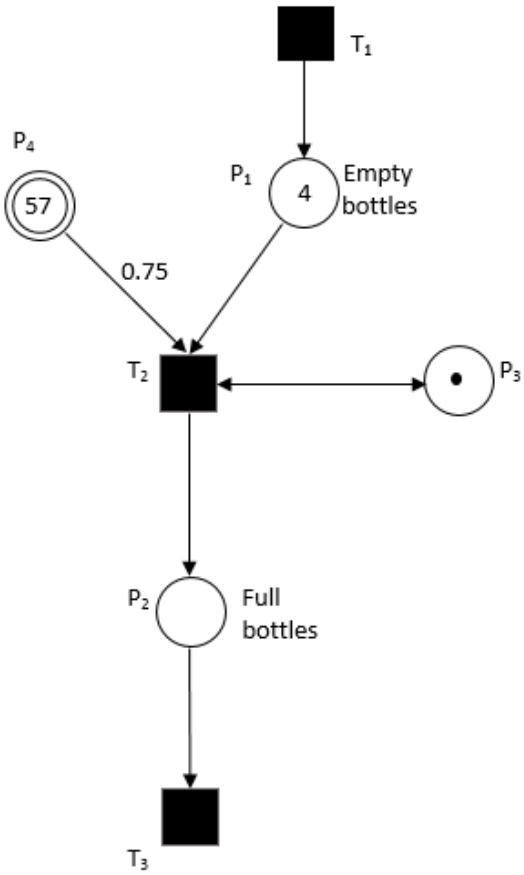

Figure 27: Petri net model of the bottling system

According to the simulation results, after the 76th step the marks in P_4 has been consumed and it is zero and empty bottles keeps on accumulating in P_1. 76 full bottles was produced. For T_2 to be enabled, at least 0.75 marks should be available in P_4. T_1 is a source transition, so there will always be token in P_1 which is required for T_2 to be enabled. The token in P_1 starts to increase after T_2 was disabled. P_3 enables or disables transition T_2 without a change to itself and also shows that various bottles cannot be produced at the same time. Table 2 shows the simulation result. This table is illustrated as histogram and graphical plot in figure 28 and 29 respectively.

Table 2: Simulation result for bottling system

Time	P4	P1	P3	P2
Initial Marking	57	4	1	0
1st Step	57	4	1	0
2nd Step	56.25	4	1	1
3rd Step	55.5	4	1	1
4th Step	54.75	4	1	1
5th Step	54	4	1	1
6th Step	53.25	4	1	1
7th Step	52.5	4	1	1
8th Step	51.75	4	1	1
9th Step	51	4	1	1
10th Step	50.25	4	1	1
11th Step	49.5	4	1	1
12th Step	48.75	4	1	1
13th Step	48	4	1	1
14th Step	47.25	4	1	1
15th Step	46.5	4	1	1
16th Step	45.75	4	1	1
17th Step	45	4	1	1
18th Step	44.25	4	1	1
19th Step	43.5	4	1	1
20th Step	42.75	4	1	1
21st Step	42	4	1	1
22nd Step	41.25	4	1	1
23rd Step	40.5	4	1	1
24th Step	39.75	4	1	1
25th Step	39	4	1	1
26th Step	38.25	4	1	1
27th Step	37.5	4	1	1
28th Step	36.75	4	1	1
29th Step	36	4	1	1
30th Step	35.25	4	1	1
31st Step	34.5	4	1	1
32nd Step	33.75	4	1	1
33rd Step	33	4	1	1
34th Step	32.25	4	1	1
35th Step	31.5	4	1	1
36th Step	30.75	4	1	1
37th Step	30	4	1	1
38th Step	29.25	4	1	1
39th Step	28.5	4	1	1
40th Step	27.75	4	1	1
41st Step	27	4	1	1
42nd Step	26.25	4	1	1
43rd Step	25.5	4	1	1
44th Step	24.75	4	1	1

Table 2 continued.

45th Step	24	4	1	1
46th Step	23.25	4	1	1
47th Step	22.5	4	1	1
48th Step	21.75	4	1	1
49th Step	21	4	1	1
50th Step	20.25	4	1	1
51st Step	19.5	4	1	1
52nd Step	18.75	4	1	1
53rd Step	18	4	1	1
54th Step	17.25	4	1	1
55th Step	16.5	4	1	1
56th Step	15.75	4	1	1
57th Step	15	4	1	1
58th Step	14.25	4	1	1
59th Step	13.5	4	1	1
60th Step	12.75	4	1	1
61st Step	12	4	1	1
62nd Step	11.25	4	1	1
63rd Step	10.5	4	1	1
64th Step	9.75	4	1	1
65th Step	9	4	1	1
66th Step	8.25	4	1	1
67th Step	7.5	4	1	1
68th Step	6.75	4	1	1
69th Step	6	4	1	1
70th Step	5.25	4	1	1
71st Step	4.5	4	1	1
72nd Step	3.75	4	1	1
73rd Step	3	4	1	1
74th Step	2.25	4	1	1

Table 2 continued.

75th Step	1.5	4	1	1
76th Step	0.75	4	1	1
77th Step	0	4	1	1
78th Step	0	5	1	0
79th Step	0	6	1	0
80th Step	0	7	1	0
81st Step	0	8	1	0
82nd Step	0	9	1	0
83rd Step	0	10	1	0
84th Step	0	11	1	0
85th Step	0	12	1	0
86th Step	0	13	1	0
87th Step	0	14	1	0
88th Step	0	15	1	0

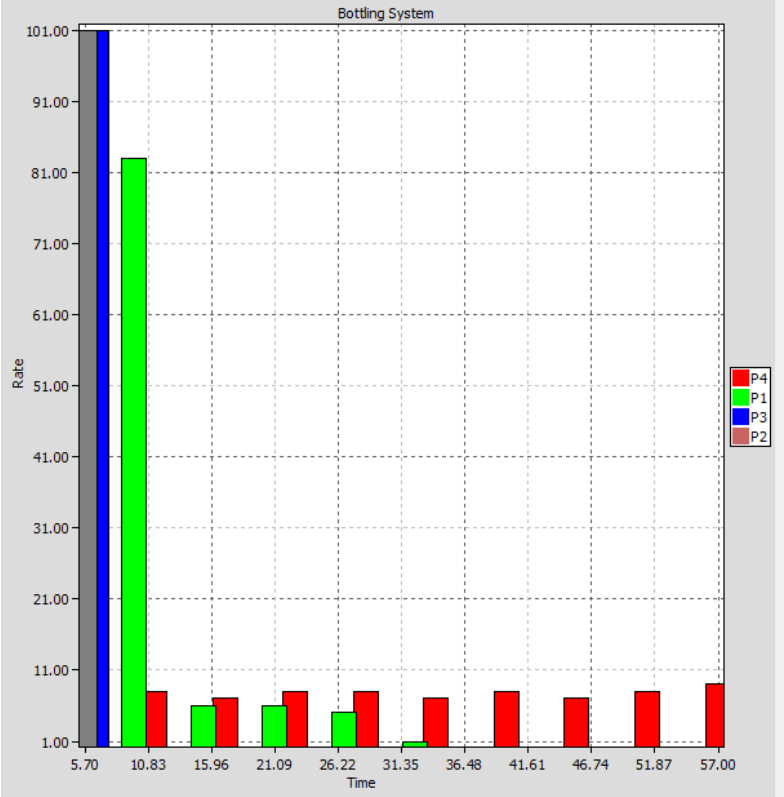

Figure 28: Histogram result of bottling system

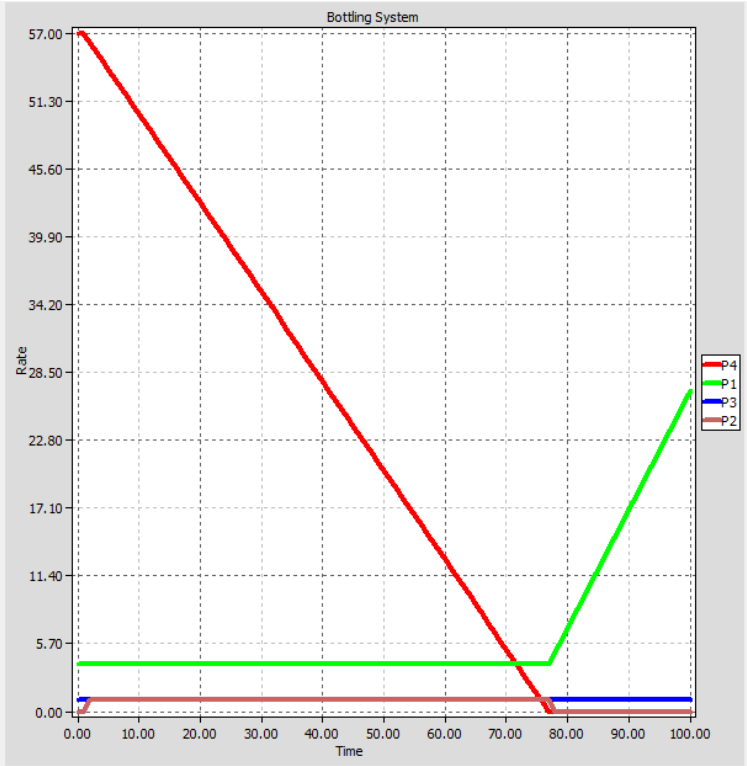

Figure 29: Graphical or xy plot of the bottling system

4.2 Production System

Figure 30 illustrates a production system where parts arrived in batches are continuously processed. T_1 (a source transition), fires and the number of tokens in P_1 keeps increasing by one. Each batch consists of 6 parts and are placed in the input buffer. The presence of a token in P_2 means that the machine is working or active. Presence of a token in P_3 means the machine has stopped working. T_2 can only be fired if there is enough space in the input buffer and is only enabled when the number of marks in P_5 is greater or equal to 6. When T_2 fires, 6 marks is passed into P_4 (the input buffer). Input buffer contains 35.26 marks in the initial marking, each batch consists of 6 pieces is passed into the input buffer, then it passes into T_5. P_2 allows or disallows T_5 (showing that discrete part influences continuous part). P_6 allows or disallows T_3, (showing that continuous part influence discrete part). The loop, P_2-T_5-P_2 indicate the

influence of discrete place on a continuous transition while the loop P_6-T_3-P_6 indicates the influence of a continuous place on a discrete place) [25]. It can be observed in the simulation results that whenever the machine stopped working (when there is a token in P_3 and no token in P_2), a marking is removed from P_6. The loops are represented with a read arc which connects P_2 to T_5 and P_6 to T_3.

The simulation results is shown in table 3. This table is also illustrated as a histogram and graph in figure 31 and 32 and respectively.

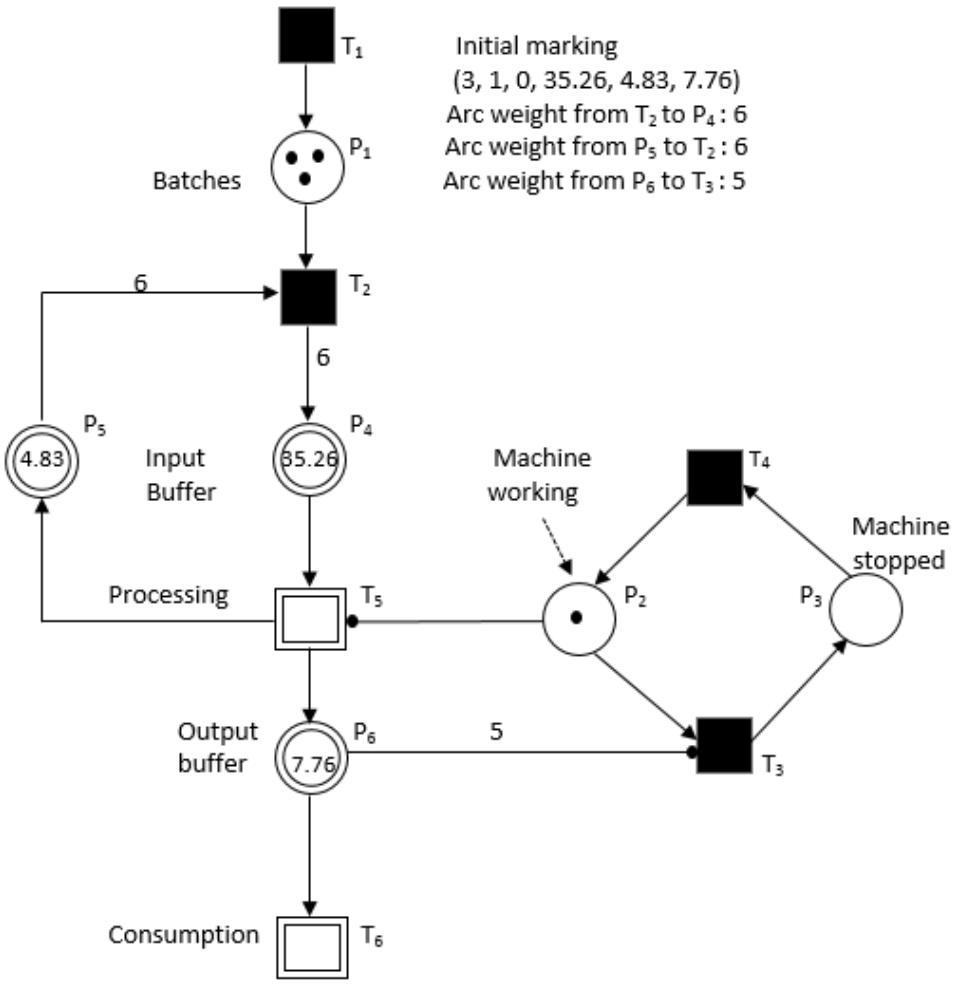

Figure 30: Petri net model of a production system

Table 3: Simulation result of production system

Time	P1	P2	P3	P4	P5	P6
Initial Step	3	1	0	35.26	4.83	7.76
1st Step	3	1	0	34.26	5.83	7.76
2nd Step	4	0	1	34.26	5.83	6.76
3rd Step	5	1	0	33.26	6.83	6.76
4th Step	5	0	1	39.26	0.83	5.76
5th Step	6	1	0	38.26	1.83	5.76
6th Step	7	0	1	38.26	1.83	4.76
7th Step	8	1	0	37.26	2.83	4.76
8th Step	9	1	0	36.26	3.83	4.76
9th Step	10	1	0	35.26	4.83	4.76
10th Step	11	1	0	34.26	5.83	4.76
11th Step	12	1	0	33.26	6.83	4.76
12th Step	12	1	0	38.26	1.83	4.76
13th Step	13	1	0	37.26	2.83	4.76
14th Step	14	1	0	36.26	3.83	4.76
15th Step	15	1	0	35.26	4.83	4.76
16th Step	16	1	0	34.26	5.83	4.76
17th Step	17	1	0	33.26	6.83	4.76
18th Step	17	1	0	38.26	1.83	4.76
19th Step	18	1	0	37.26	2.83	4.76
20th Step	19	1	0	36.26	3.83	4.76
21st Step	20	1	0	35.26	4.83	4.76
22nd Step	21	1	0	34.26	5.83	4.76
23rd Step	22	1	0	33.26	6.83	4.76
24th Step	22	1	0	38.26	1.83	4.76
25th Step	23	1	0	37.26	2.83	4.76
26th Step	24	1	0	36.26	3.83	4.76
27th Step	25	1	0	35.26	4.83	4.76
28th Step	26	1	0	34.26	5.83	4.76
29th Step	27	1	0	33.26	6.83	4.76
30th Step	27	1	0	38.26	1.83	4.76
31st Step	28	1	0	37.26	2.83	4.76
32nd Step	29	1	0	36.26	3.83	4.76
33rd Step	30	1	0	35.26	4.83	4.76
34th Step	31	1	0	34.26	5.83	4.76
35th Step	32	1	0	33.26	6.83	4.76
36th Step	32	1	0	38.26	1.83	4.76
37th Step	33	1	0	37.26	2.83	4.76
38th Step	34	1	0	36.26	3.83	4.76
39th Step	35	1	0	35.26	4.83	4.76
40th Step	36	1	0	34.26	5.83	4.76
41st Step	37	1	0	33.26	6.83	4.76
42nd Step	37	1	0	38.26	1.83	4.76
43rd Step	38	1	0	37.26	2.83	4.76
44th Step	39	1	0	36.26	3.83	4.76

Table 3 continued

45th Step	40	1	0	35.26	4.83	4.76
46th Step	41	1	0	34.26	5.83	4.76
47th Step	42	1	0	33.26	6.83	4.76
48th Step	42	1	0	38.26	1.83	4.76
49th Step	43	1	0	37.26	2.83	4.76
50th Step	44	1	0	36.26	3.83	4.76
51st Step	45	1	0	35.26	4.83	4.76
52nd Step	46	1	0	34.26	5.83	4.76
53rd Step	47	1	0	33.26	6.83	4.76
54th Step	47	1	0	38.26	1.83	4.76
55th Step	48	1	0	37.26	2.83	4.76
56th Step	49	1	0	36.26	3.83	4.76
57th Step	50	1	0	35.26	4.83	4.76
58th Step	51	1	0	34.26	5.83	4.76
59th Step	52	1	0	33.26	6.83	4.76
60th Step	52	1	0	38.26	1.83	4.76
61st Step	53	1	0	37.26	2.83	4.76
62nd Step	54	1	0	36.26	3.83	4.76
63rd Step	55	1	0	35.26	4.83	4.76
64th Step	56	1	0	34.26	5.83	4.76
65th Step	57	1	0	33.26	6.83	4.76
66th Step	57	1	0	38.26	1.83	4.76
67th Step	58	1	0	37.26	2.83	4.76
68th Step	59	1	0	36.26	3.83	4.76
69th Step	60	1	0	35.26	4.83	4.76
70th Step	61	1	0	34.26	5.83	4.76
71st Step	62	1	0	33.26	6.83	4.76
72nd Step	62	1	0	38.26	1.83	4.76
73rd Step	63	1	0	37.26	2.83	4.76
74th Step	64	1	0	36.26	3.83	4.76
75th Step	65	1	0	35.26	4.83	4.76
76th Step	66	1	0	34.26	5.83	4.76
77th Step	67	1	0	33.26	6.83	4.76
78th Step	67	1	0	38.26	1.83	4.76
79th Step	68	1	0	37.26	2.83	4.76
80th Step	69	1	0	36.26	3.83	4.76
81st Step	70	1	0	35.26	4.83	4.76
82nd Step	71	1	0	34.26	5.83	4.76
83rd Step	72	1	0	33.26	6.83	4.76
84th Step	72	1	0	38.26	1.83	4.76
85th Step	73	1	0	37.26	2.83	4.76
86th Step	74	1	0	36.26	3.83	4.76
87th Step	75	1	0	35.26	4.83	4.76
88th Step	76	1	0	34.26	5.83	4.76
89th Step	77	1	0	33.26	6.83	4.76
90th Step	77	1	0	38.26	1.83	4.76
91st Step	78	1	0	37.26	2.83	4.76

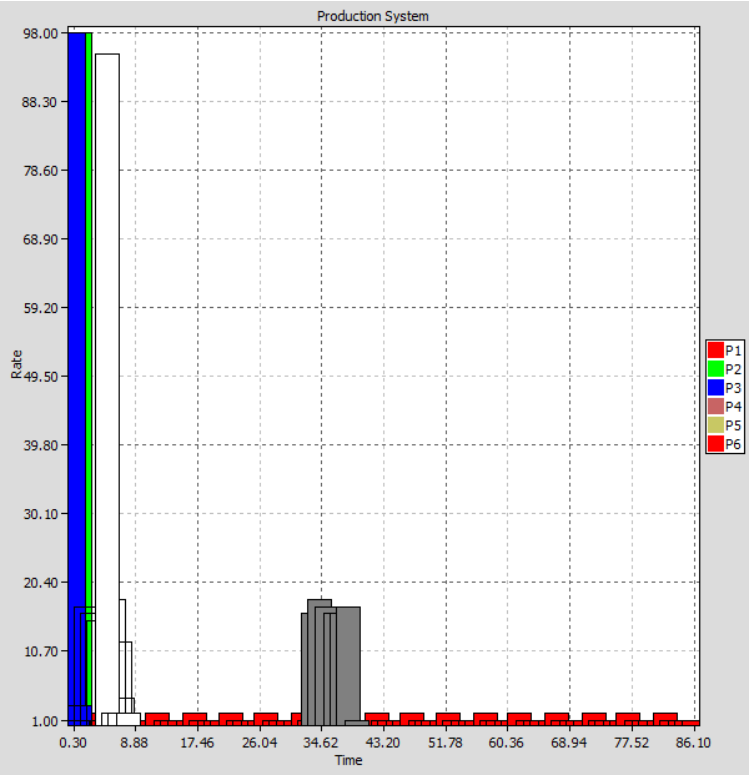

Figure 31: Histogram result of production system

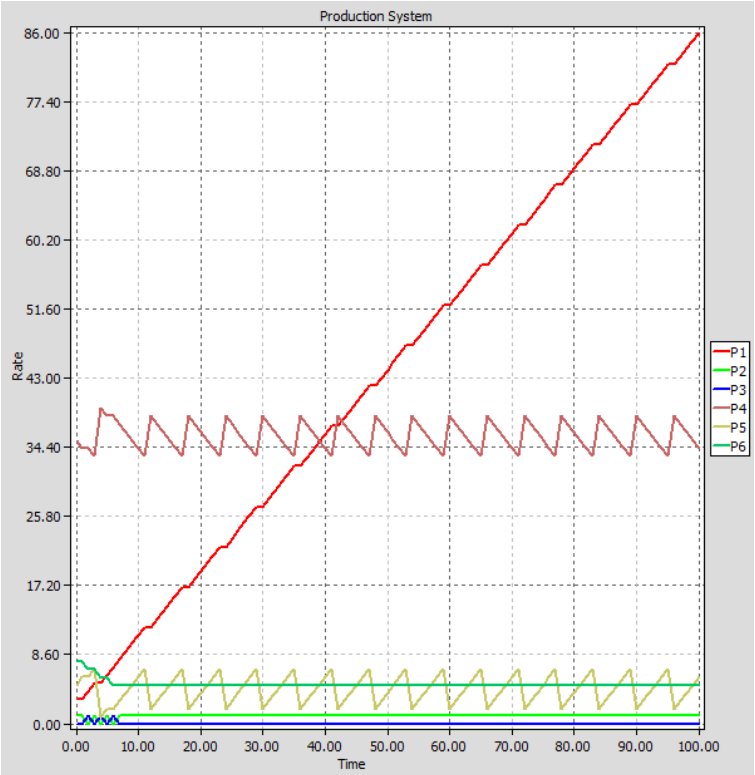

Figure 32: Graphical or xy-plot result of production system

4.3 Olive Soap Production System

The olive soap production system is illustrated in figure 33. This case study is taken from an unpublished book of Prof. Dr. Rza Bashirov. I remain grateful to him for giving me access to his work, thus adding value to this study.

Initially, there are no marks in any of the places. For P_1, 50 tokens is consumed from T_1 in the first step. The difference between the first step and second step in P_1 is 45. This is equivalent to the difference in the pre and post arc weight. This is because when T_1 fires it deposit 50 tokens and when T_2 fires, 5 tokens are removed. This implies that the difference between the steps keeps increasing by 45.

For T_3 to be enabled, there has to be at least 0.354 marks in P_2, 1.179 marks in P3 and 0.136 marks in P_4. Since the arc weight from T_2 to P_3 is 1, there has to be two marks in P_3 for T_3 to be enabled, so that 1.179 marks can be consumed by T_3 from P_3, the transition occurred at the 5^{th} step. At the 5^{th} step a token was deposited to P_5 since T_3 is enabled. T_3 is disabled when the number of marks in P3 is not up to 1.179. This occurs in the 9^{th}, 16^{th}, 22^{nd}, 29^{th}, and 35^{th} steps. In the 10^{th}, 17^{th}, 23^{rd}, 30^{th} and 36^{th} step, P_5 has no token deposited in it (no soap production), the number of tokens in P_2, P_3 and P_6 is increased by exactly one mark.

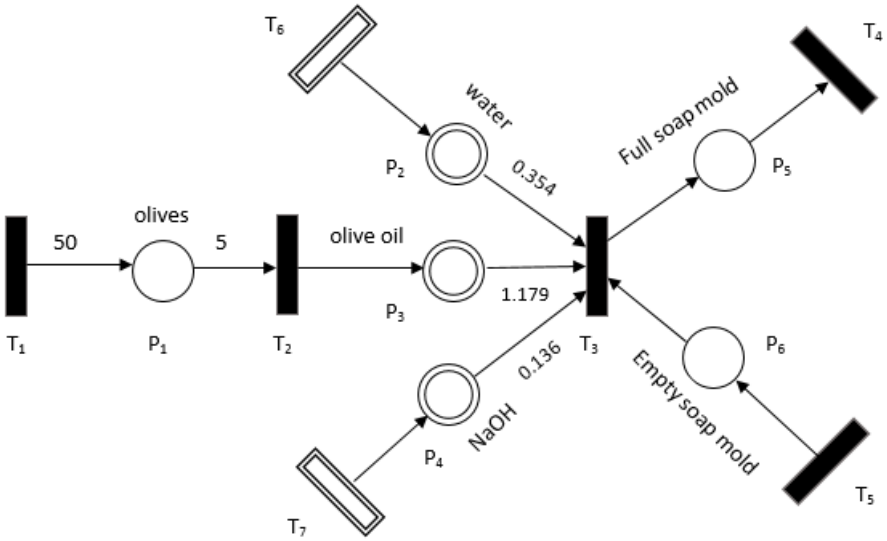

Figure 33: Petri net model of the olive soap production system

The simulation results obtained is shown in table 4. This table is also illustrated as histogram and graphical chart in figure 34 and 35 respectively.

Table 4: Simulation result of olive soap production system

Time	P1	P2	P3	P4	P5	P6
Initial Marking	0	0	0	0	0	0
1st Step	0	1	0	1	0	0
2nd Step	50	2	0	2	0	1
3rd Step	95	3	1	3	0	2
4th Step	140	4	2	4	0	3
5th Step	185	4.646	1.821	4.864	1	3
6th Step	230	5.292	1.642	5.728	1	3
7th Step	275	5.938	1.463	6.592	1	3
8th Step	320	6.584	1.284	7.456	1	3
9th Step	365	7.23	1.105	8.32	1	3
10th Step	410	8.23	2.105	9.32	0	4
11th Step	455	8.876	1.926	10.184	1	4
12th Step	500	9.522	1.747	11.048	1	4
13th Step	545	10.168	1.568	11.912	1	4
14th Step	590	10.814	1.389	12.776	1	4
15th Step	635	11.46	1.21	13.64	1	4
16th Step	680	12.106	1.031	14.504	1	4
17th Step	725	13.106	2.031	15.504	0	5
18th Step	770	13.752	1.852	16.368	1	5
19th Step	815	14.398	1.673	17.232	1	5
20th Step	860	15.044	1.494	18.096	1	5
21st Step	905	15.69	1.315	18.96	1	5
22nd Step	950	16.336	1.136	19.824	1	5
23rd Step	995	17.336	2.136	20.824	0	6
24th Step	1040	17.982	1.957	21.688	1	6
25th Step	1085	18.628	1.778	22.552	1	6
26th Step	1130	19.274	1.599	23.416	1	6
27th Step	1175	19.92	1.42	24.28	1	6
28th Step	1220	20.566	1.241	25.144	1	6
29th Step	1265	21.212	1.062	26.008	1	6
30th Step	1310	22.212	2.062	27.008	0	7
31st Step	1355	22.858	1.883	27.872	1	7
32nd Step	1400	23.504	1.704	28.736	1	7
33rd Step	1445	24.15	1.525	29.6	1	7
34th Step	1490	24.796	1.346	30.464	1	7
35th Step	1535	25.442	1.167	31.328	1	7
36th Step	1580	26.442	2.167	32.328	0	8
37th Step	1625	27.088	1.988	33.192	1	8
38th Step	1670	27.734	1.809	34.056	1	8
39th Step	1715	28.38	1.63	34.92	1	8
40th Step	1760	29.026	1.451	35.784	1	8
41st Step	1805	29.672	1.272	36.648	1	8
42nd Step	1850	30.318	1.093	37.512	1	8
43rd Step	1895	31.318	2.093	38.512	0	9
44th Step	1940	31.964	1.914	39.376	1	9

Table 4 continued

45th Step	1985	32.61	1.735	40.24	1	9
46th Step	2030	33.256	1.556	41.104	1	9
47th Step	2075	33.902	1.377	41.968	1	9
48th Step	2120	34.548	1.198	42.832	1	9
49th Step	2165	35.194	1.019	43.696	1	9
50th Step	2210	36.194	2.019	44.696	0	10
51st Step	2255	36.84	1.84	45.56	1	10
52nd Step	2300	37.486	1.661	46.424	1	10
53rd Step	2345	38.132	1.482	47.288	1	10
54th Step	2390	38.778	1.303	48.152	1	10
55th Step	2435	39.424	1.124	49.016	1	10
56th Step	2480	40.424	2.124	50.016	0	11
57th Step	2525	41.07	1.945	50.88	1	11
58th Step	2570	41.716	1.766	51.744	1	11
59th Step	2615	42.362	1.587	52.608	1	11
60th Step	2660	43.008	1.408	53.472	1	11
61st Step	2705	43.654	1.229	54.336	1	11
62nd Step	2750	44.3	1.05	55.2	1	11
63rd Step	2795	45.3	2.05	56.2	0	12
64th Step	2840	45.946	1.871	57.064	1	12
65th Step	2885	46.592	1.692	57.928	1	12
66th Step	2930	47.238	1.513	58.792	1	12
67th Step	2975	47.884	1.334	59.656	1	12
68th Step	3020	48.53	1.155	60.52	1	12
69th Step	3065	49.53	2.155	61.52	0	13
70th Step	3110	50.176	1.976	62.384	1	13
71st Step	3155	50.822	1.797	63.248	1	13
72nd Step	3200	51.468	1.618	64.112	1	13
73rd Step	3245	52.114	1.439	64.976	1	13
74th Step	3290	52.76	1.26	65.84	1	13
75th Step	3335	53.406	1.081	66.704	1	13
76th Step	3380	54.406	2.081	67.704	0	14
77th Step	3425	55.052	1.902	68.568	1	14
78th Step	3470	55.698	1.723	69.432	1	14
79th Step	3515	56.344	1.544	70.296	1	14
80th Step	3560	56.99	1.365	71.16	1	14
81st Step	3605	57.636	1.186	72.024	1	14
82nd Step	3650	58.282	1.007	72.888	1	14
83rd Step	3695	59.282	2.007	73.888	0	15
84th Step	3740	59.928	1.828	74.752	1	15
85th Step	3785	60.574	1.649	75.616	1	15
86th Step	3830	61.22	1.47	76.48	1	15
87th Step	3875	61.866	1.291	77.344	1	15
88th Step	3920	62.512	1.112	78.208	1	15
89th Step	3965	63.512	2.112	79.208	0	16
90th Step	4010	64.158	1.933	80.072	1	16

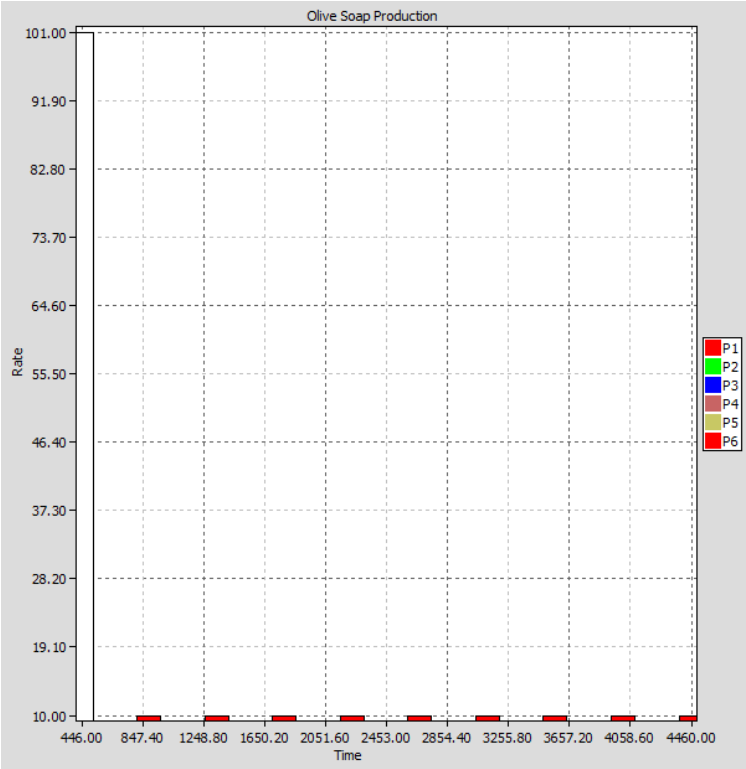

Figure 34: Histogram result of olive soap production system

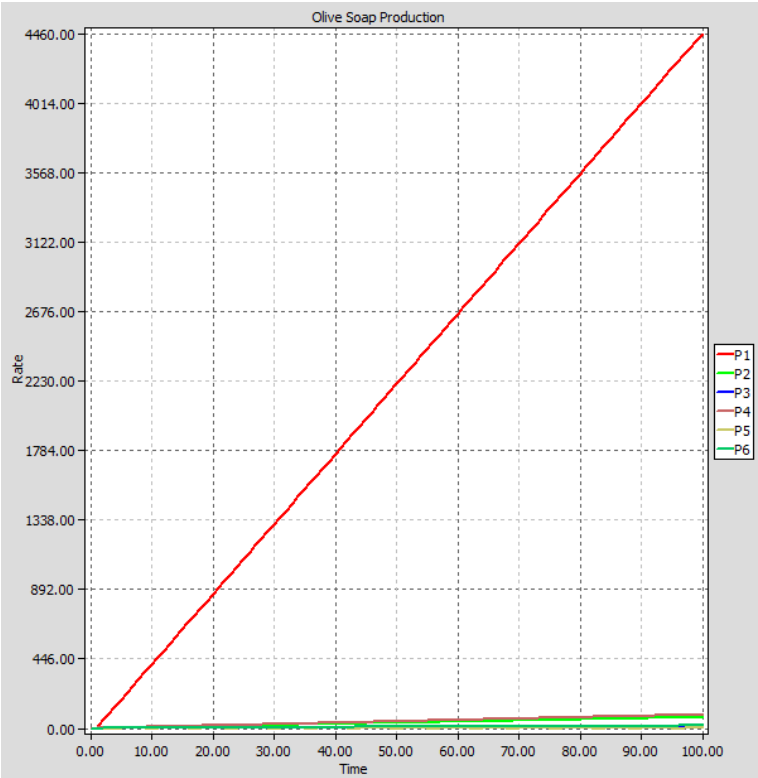

Figure 35: Graphical or xy-plot of olive soap production system

Chapter 5

CONCLUSION

This thesis aims at studying the inter-relationship that exists between discrete and continuous Petri nets components. It was noted that a growing demand exists in the application of Petri nets in the fields of biomedicine, biochemistry, molecular biology, and systems biology thus creating a wider scope in their applications.

The thesis also as a part of its findings noted that Petri net involved modelling and simulation which gives researchers a broader view of the biological processes thereby giving a detailed and better comprehension of the nature of biological systems.

The main outcomes of this work are as follows:

- Petri nets can be modelled using different simulation software, some of which were discussed earlier.
- Snoopy software was used in modelling the case studies and we were able to generate results in Tabular, xy-plot and histogram forms.
- Rate of continuous transition using mass action was calculated to be approximately 0.63.
- Standard arc can connect like places and transitions together.
- Standard arc can connect continuous place and discrete transition, but not discrete place and continuous transition.
- To connect discrete place and continuous transition, the read arc is used which acts like a loop. This is used in modelling of the production system.

- Continuous places accepts real number values while discrete place accepts only integer values.

With the various software examined, one can deduced that HPsim is suitable for beginners although it has limited place capacity, while with HPSim, knowledge of MATLAB is required since its GUI is embedded in this environment. HiQPN, another software tool, is suitable for large and complex model. PetriSim on the other hand, encourages the use of Object oriented programming which requires a good knowledge of pascal Programming language. HISim helps to solve accuracy issue simply because its place can take on both positive and negative real number values.

The work concludes that Petri nets can be used at modelling simple and complex systems. It also shows that snoopy can be used as a simulation software for Petri net construction simulation and analysis.

REFERENCES

[1] "Carl Adam Petri," [Online]. Available: https://en.wikipedia.org/wiki/Carl_Adam_Petri. [Accessed 2 July 2018].

[2] "Snoopy," [Online]. Available: http://www-dssz.informatik.tu-cottbus.de/DSSZ/Software/Snoopy. [Accessed 03 March 2018].

[3] R. Bashirov, "Comp 544 Lecture Notes," 2017.

[4] "Petri Nets," [Online]. Available: https://www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/pnfaq.html. [Accessed 25 July 2018].

[5] W. Zuberek, "Timed Petri nets, Definitions, Properties, and applications," vol. IV, no. 33, pp. 627-644, 1991.

[6] C. Rohr, M. Heiner, W. Marwan, "Petri nets in Snoopy: A unifying frame-work for the graphical display, computational modelling, and simulation of bacterial regulatory networks," 2010.

[7] T. Karanfiller, R. Bashirov, " Exploiting Petri Nets to Reduce Switch Crosstalk and Path-Dependent-Loss in Optical Interconnection Networks," January 2012.

[8] D. Gilbert, R. Donaldson, H. Monika, "Petri Nets for Systems and Synthetic Biology," pp. 215-264, 2008.

[9] F. Dolma, R. Basirov, "Implementing Petri Nets for Modelling and Simulation in Biosciences," January 2012.

[10] J. Ackermann, I. Koch, " Modeling in Systems Biology: The Petri Nets Approach," *Springer*, pp. 153-178, 2011.

[11] A. Krivdic, "Mathematical and Computational Models of Cell Cycle in Higher Eukaryotes," September 2015.

[12] "Chemical equilibrum," [Online]. Available: https://en.wikipedia.org/wiki/Chemical_equilibrium. [Accessed 10 July 2018].

[13] "Steady State(chemistry)," [Online]. Available: https://en.wikipedia.org/wiki/Steady_state_(chemistry). [Accessed 10 July 2018].

[14] "IUPAC Gold Book definition of steady state," [Online]. Available: https://goldbook.iupac.org. [Accessed 12 July 2018].

[15] "Steady State," [Online]. Available: https://en.wikipedia.org/wiki/Steady_state_(chemistry). [Accessed 11 July 2018].

[16] "HPSim," [Online]. Available: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/hpsim.html. [Accessed 13 July 2018].

[17] "SimHPN," [Online]. Available: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/simhpn.html. [Accessed 10 July 2018].

[18] P. Buchholz. P. Kemper, F. Bause, "QPN-Tool for the Specification and Analysis of Hierarchically Combined Queueing Petri Nets. Quantitative Evaluation of Computing and Communication Systems," in *8th International Conference on Modelling Techniques*, September 1995.

[19] "HiQPN-Tool," [Online]. Available: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db/hiqpntool.html. [Accessed 14 July 2018].

[20] P. Mynampati, N. Tiwari, "Experiences of using LQN and QPN tools for performance modeling of a J2EE Application".

[21] "Petri Nets Tools Database," [Online]. Available: https://www.informatik.uni-hamburg.de/TGI/PetriNets/tools/db.html. [Accessed 25 July 2018].

[22] "PetriSim," [Online]. Available: http://staff.um.edu.mt/jskl1/petrisim/. [Accessed 10 July 2018].

[23] A. Alberto, "A specification of a hybrid Petri net semantics for the HISim simulator," March, 2009.

[24] R. Ritcher, M. Schwarick, C. Rohr, M. Heiner, "Snoopy- A tool to design and Execute Graph-Based Formalisms (Extended Version)," *Brandenburg University of Technology at Cottbus, Germany.*

[25] H. Alla, R. David, "Discrete, Continuous, and Hybrid Petri Nets", 2005.