Graduate School Form
30 Updated 12/26/2015

PURDUE UNIVERSITY
GRADUATE SCHOOL
Thesis/Dissertation Acceptance

This is to certify that the thesis/dissertation prepared

By Archana Eadara

Entitled

MODELING, ANALYSIS, AND SIMULATION OF MUZIMA FINGERPRINT MODULE BASED ON ORDINARY AND TIME PETRI NETS

For the degree of Master of Science in Electrical and Computer Engineering

Is approved by the final examining committee:

Lingxi Li
Chair

Brian King

Maher Rizkalla

To the best of my knowledge and as understood by the student in the Thesis/Dissertation Agreement, Publication Delay, and Certification Disclaimer (Graduate School Form 32), this thesis/dissertation adheres to the provisions of Purdue University's "Policy of Integrity in Research" and the use of copyright material.

Approved by Major Professor(s): Lingxi Li

Approved by: Brian King 4/21/2016

Head of the Departmental Graduate Program Date

MODELING, ANALYSIS, AND SIMULATION OF MUZIMA FINGERPRINT MODULE BASED ON ORDINARY AND TIME PETRI NETS

A Thesis

Submitted to the Faculty

of

Purdue University

by

Archana Eadara

In Partial Fulfillment of the

Requirements for the Degree

of

Master of Science in Electrical and Computer Engineering

May 2016

Purdue University

Indianapolis, Indiana

To all those who supported me achieve this.

ACKNOWLEDGMENTS

I would like to thank God for blessing me with the confidence and strength which led to the successful completion of my thesis.

Also, I would like to sincerely thank my major professor, Dr. Lingxi Li for all his guidance and encouragement throughout the research and writing process. I would also like to thank Dr. Brian King and Dr. Maher Rizkalla for serving on my dissertation committee. I am very grateful for all of your support since I started pursuing Masters degree. I would also like to acknowledge Sherrie Tucker and all other ECE Department faculty and staff for their assistance without which I would not have achieved this.

Lastly, I would like to thank Muzima Team for their guidance and help in the development of Muzima Fingerprint Module during my internship with them.

My family and friends kept me motivated the entire time. Your endless love and support means a lot to me.

Thank you,

Archana

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Eadara, Archana. M.S.E.C.E., Purdue University, May 2016. Modeling, Analysis, and Simulation of Muzima Fingerprint Module Based on Ordinary and Time Petri Nets. Major Professor: Lingxi Li.

In the healthcare industry, several modern patient identification and patient matching systems have been introduced. Most of these implement patient identification by their first, middle and last names. They also use Social Security Number and other similar national identifiers. These methods may not work for many developing and underdeveloped countries where identifying a patient is a challenge with highly redundant and interchangeable first and last names of the patient, this is aggravated by the absence of a national identification system. In order to make the patient identification more efficient, Muzima, an interface of OpenMRS (Open source medical records system) introduced an additional identifier, fingerprint, through a module to the system. Ordinary and Time Petri nets are used to analyze this module. Chapter 1 introduces Muzima fingerprint module and describes the workflow of this interface followed by the related work, importance and applications of Petri nets. Chapter 2 introduces Ordinary and Time Petri nets using examples. Chapter 3 discusses about the mathematical modeling of the Muzima Fingerprint module using Petri nets. Chapter 4 explains the qualitative and quantitative analysis done on the Muzima fingerprint module. Chapter 5 discusses about the programming and simulation done to prove the theoretical results obtained. Chapter 6 provides the conclusion and future work for the thesis.

1. INTRODUCTION

In the healthcare sector around the world, the use of Electronic Medical Records (EMRs) is being encouraged due to the overwhelming digital evolution. One of the most important advantages of using EMRs is the access to a patient record just a click away. For successful operation of this feature, efficient patient identification and patient matching algorithms are needed. For reasons such as lack of national identification systems in many developing and underdeveloped nations and various patient data entry issues, it is a challenge to access a specific patient record [1]. To maintain this feature, there is a lot of ongoing research to improve these patient identification and patient matching algorithms.

1.1 Patient Identification and Patient Matching

Healthcare organizations employ a variety of methods that identify patients at registration and to access patient records. These methods vary from a simple attribute match to highly advanced algorithms. These attributes may include first name, middle name, last name, date of birth, medical record number (MRN), national identifier and other similar data. Such attributes are needed to identify and discern a patient record among different data sources such as laboratory, pharmacy and different hospitals.

An efficient patient identification and patient matching method is needed to identify and retrieve a patient record from the system to provide immediate or follow-up care. In a survey conducted by College of Healthcare Information Management Executives (CHIME), 20% of the Chief Information Officers (CIOs) pointed out that a minimum one patient suffered antagonistic condition in the past year due to mismatch of patient records [2]. Another study indicates the risk of misallocation of laboratory

observation reports due to the presence of duplicate records [3]. This may result into a threat to patient safety due to multiple clinical workflows. Accurate identification and matching of patient record is needed as data is stored and transported digitally among several sources and recipients.

All humans make mistakes. There are many chances of entering patient data incorrectly into the EMR. For instance, incorrect birthdate, misspelled names, redundant names, interchangeable first and last names. This may lead to the creation of duplicate patient records (multiple records created for the same patient). In such cases, We cannot depend upon the MRN (unique identifier generated by the EMR) assigned to the patient record. All these factors make patient identification and patient matching more complicated. Many EMRs overcome these issues by relying on a unique identifier such as a national identifier. Many developing and underdeveloped nations do not yet implement a national identification system. In the absence of such an identifier it is a challenge to identify a patient record.

1.2 Muzima Fingerprint module

OpenMRS (Open source medical records system) is a free, open source health IT software created by volunteers from around the globe. It has evolved into a global health IT solution and has implementations in over 80 countries along with translations into various languages. Presently, OpenMRS is in use in 1,149 centers around the world improving medical care for over 5.1 million patients [4].

Emerging technologies like Biometrics act as an enhancement to the existing patient identification and patient matching algorithms. Muzima, an interface of OpenMRS introduced the use of patient fingerprint as an additional attribute to assist these algorithms. Muzima fingerprint module works on patient identification by first scanning the patient finger and using the fingerprint as the search input. This module leverages a third party technology (fingerprint applet) to identify fingerprints. This technology requires acquiring licenses to use its service. This technology verifies if the

request is coming from a licensed source each time we access it to identify a fingerprint. The workflow of the Muzima Fingerprint Module (Fig. 1.1) can be explained

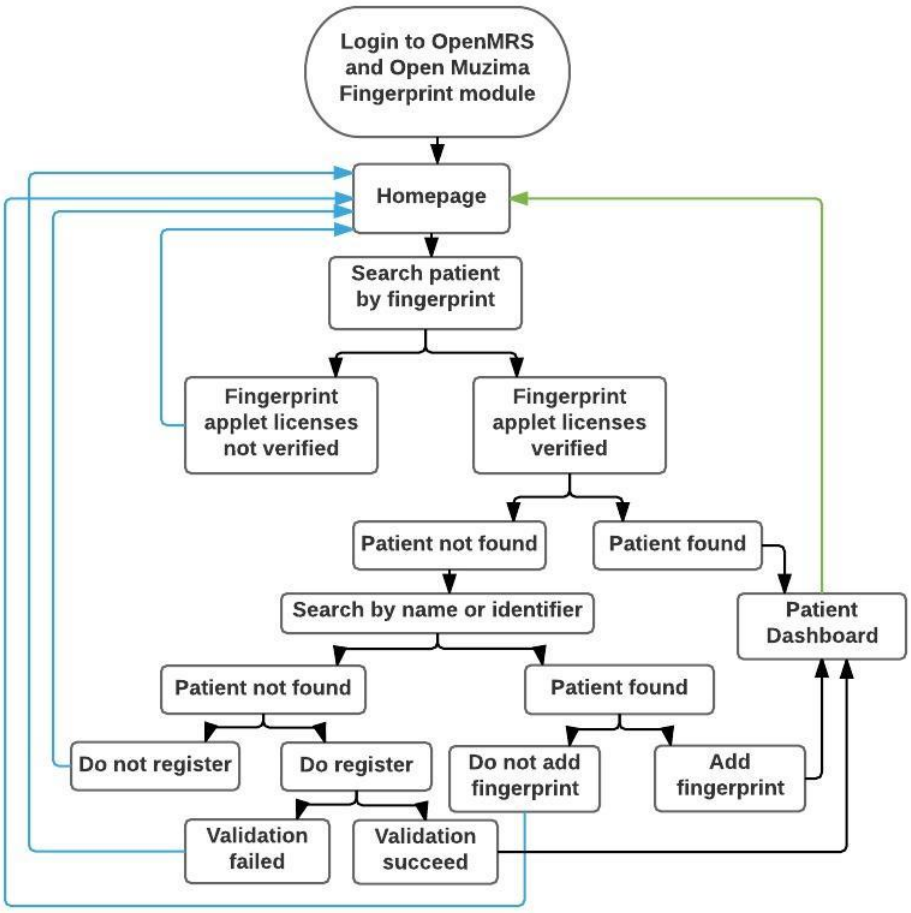

Fig. 1.1. Brief workflow of Muzima fingerprint module

as following:

1. Login into OpenMRS.
2. After successful login, locate Muzima Fingerprint module and click to launch it.
3. In the module homepage, click on "Search by fingerprint" button to find patient using patient's fingerprint as search input.

4. This will launch the fingerprint applet and allow scanning of the finger.

5. If the licenses of the fingerprint applet are verified successfully, scanning continues otherwise scanning stops and returns back to home page with an error message to the user.

6. In case of successful verification of licenses, the search will return a patient record if the patient is already registered in the EMR with his/her fingerprint or returns a message "No patient found with this fingerprint".

7. When no patient is found with the fingerprint, either the patient is not registered in the EMR or the patient record is not updated with the fingerprint data. In this case, we can search by patient name or identifier.

8. If the patient is still not found, an option is given to register the patient. If a patient is found, then an option is given to add fingerprint.

9. Whenever a patient record is returned, clicking on the patient record takes to the patient dashboard.

1.3 Applications of Petri Nets in Healthcare: Previous related work

Petri net is a graphical mathematical modeling tool applicable for describing and studying various systems such as distributed, parallel and deterministic. They also help communicate the content visually similar to other graphical tools such as flow charts, block diagrams and networks.

Petri nets are currently used for optimal modeling in various industry areas including healthcare. In any healthcare system, a patient is diagnosed of disease and treated by carrying out the procedures which follow a medical protocol. [5] proposes that for nearly all of the diseases, a medical protocol can be modeled by using a Petri net. The different resources that are needed by the medical protocol are represented by places in the Petri net. The available instances of a specific resource are represented by tokens in that respective place.

Petri nets are used in another research to analyze and construct the workflow of tele-healthcare (healthcare services built on the internet). The existence of states and token based flow of Petri nets make the resource transfer process among different states very clearly, helping the care providers ensure the right care actions are given at the right time [6].

Medical data privacy invasion is one of the major concerns in the E-healthcare setting due to the increasing number of cases of unauthorized patient medical data access. In order to improve patient medical data security, [7] proposes an RFID healthcare framework. The workflow of this proposed system verifies boundedness, liveness, reachability, and reversibility by applying Petri nets. An advanced Petri net model has been used to design a framework for trust development between trust factors and healthcare behaviors in a global healthcare setting [8].

A high-level Petri net called Coloured Petri net (CPN) has been applied to a case study conducted in a mental healthcare institute about enhancing the intake process (reduction of service time and flow time). Here, CPNs are used to perform simulations for statistical analysis in order to compare the performance of different models [9]. Another application of high-level Petri nets discusses a change management framework for a healthcare service platform. This study introduces a combination of the Petri net model (to represent system requirement changes) and redesignable Petri nets model (to react to these changes) [10].

Timed Petri nets are another extension of Petri nets where the transitions can be timed. [11] presents an application of Timed Petri nets in a healthcare setting. This study uses hierarchical Timed Petri nets to reduce the complexity of the healthcare process model and adds time constraints to the process. This method helps the designers to test and verify timing constraint satisfiability.

The application of Petri Nets to analyze and improve various frameworks and workflows in healthcare is the major inspiration of this thesis.

1.4 Thesis contributions

The wide range of applications of Petri nets in the healthcare industry was the motivation behind developing a generic mathematical and graphical model for Muzima Fingerprint module in OpenMRS, on which analysis and simulation is performed.

The contributions of this thesis work can be stated as follows:

1. Proposed a mathematical and graphical model for the workflow of the Muzima Fingerprint module using Ordinary Petri nets.

2. Extended the Petri net model in order to include timing details for transitions using Time Petri Nets.

3. Developed an algorithm and a generic MATLAB program to analyze and verify the reachability properties of the Petri net model developed.

4. Performed retrospective Petri net analysis (on structural and dynamic properties) taking Muzima Fingerprint module as an example. This can be used as a template for analyzing the behavior of any software workflow.

    - For retrospective analysis, we can identify issues and improve the current workflow.
    - For prospective analysis, we can avoid potential errors in the envisioned workflow.

1.5 Thesis organization

This thesis is structured into six chapters. Chapter 1 gives an overview about the Patient Identification and Patient Matching methods employed by healthcare organizations and the need for regular development of these methods. It explains the workflow of the Muzima Fingerprint module. It also provides previous literature work related to applications of Petri nets in healthcare. Chapter 2 introduces the mathematical and graphical modeling tool employed in this thesis i.e. Petri nets

(Ordinary Petri net and Time Petri net) and demonstrates their properties using examples. The objective of this chapter is to provide the reader a basic background about Petri nets for understanding the next chapters. Chapter 3 mainly discusses about the mathematical modeling of the Muzima Fingerprint module using Petri nets. Then, it continues to explain the extension of this Petri net model to include timing details for transitions. Chapter 4 explains the qualitative and quantitative analysis done on the Petri Net model developed. It also discusses about the algorithm developed for reachability tree method of analysis. Chapter 5 discusses about the MATLAB program developed and simulation to prove the theoretical results obtained for qualitative and quantitative analysis of the system. Chapter 6 concludes and discusses future work of this thesis.

2. INTRODUCTION TO PETRI NETS

Petri net is one of the various mathematical and graphical modeling tools for representing distributed systems. The theory of Petri net was invented by Carl Adam Petri in August 1939 [12]. Petri nets have finite states with infinite memory, thus overcoming the finite memory limitations of the Finite State Machines (FSMs).

2.1 Definition, Structure, Syntax and Terminology

A Petri net graph can be defined as a weighted directed graph described as a 4-tuple function:

$$N = (P, T, A, W)$$

The elements of the function can be described as:

P a finite set of places(represent conditions under which events can occur in a discrete event system) which are drawn as circles.

T a finite set of transitions(represent events driving a discrete event system) which are drawn as bars.

A defined as $(P \times T) \cup (T \times P)$ is a set of arcs which connect places to transitions or transitions to places. There will be no arcs connecting any two places or any two transitions.

W defined as $A \to \mathbb{N}$ (natural numbers) is the weight function that assigns each arc a natural number.

A simple Petri net graph is shown in Fig. 2.1.

From the graph we can derive,

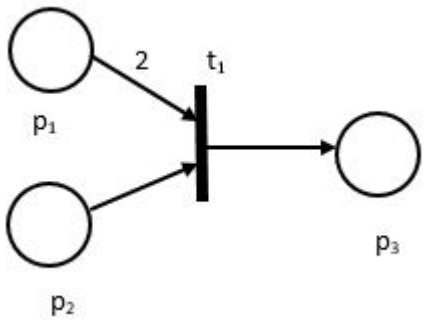

Fig. 2.1. An Example of Petri net graph

$P = \{p_1, p_2, p_3\}$,
$T = \{t_1\}$,
$A = \{(p_1, t_1), (p_2, t_1), (t_1, p_3)\}$,
$W(p_1, t_1) = 2$, $W(p_2, t_1) = 1$ and $W(t_1, p_3) = 1$ (the default weight of an arc is 1).

We use,

$I(t_j)$ to denote the set of input places for transition t_j.

$$I(t_j) = \{p_i \mid p_i \in P, (p_i, t_j) \in A\}$$

$O(t_j)$ to denote the set of output places for transition t_j.

$$O(t_j) = \{p_i \mid p_i \in P, (t_j, p_i) \in A\}$$

$I(p_i)$ to denote the set of input transitions to place p_i.

$$I(p_i) = \{t_j \mid t_j \in T, (t_j, p_i) \in A\}$$

$O(p_i)$ to denote the set of output transitions to place p_i.

$$O(p_i) = \{t_j \mid t_j \in T, (p_i, t_j) \in A\}$$

Consider the example Petri net graph shown in Fig. 2.1. From the Petri net we can say,

$$I(t_1) = \{p_1, p_2\}$$
$$O(t_1) = \{p_3\}$$

$$I(p_1) = \emptyset$$
$$O(p_1) = \{t_1\}$$
$$I(p_2) = \emptyset$$
$$O(p_2) = \{t_1\}$$
$$I(p_3) = \{t_1\}$$
$$O(p_3) = \emptyset$$

2.2 Petri net marking

As mentioned in section 2.1, transitions of a Petri net represent events that occur in a discrete event system and places of a Petri net represent conditions under which these events can take place. By assigning tokens (drawn as black dots in a place) to places we can indicate if these conditions are met. A Petri net marking M is defined as, the way in which tokens are assigned to places [13]. Marking M can be represented in a function as:

$$M : P \to \mathbb{W} = \{0, 1, 2, \cdots\}$$

Marking M is a $n \times 1$ column vector where n represents the number of places in a Petri net.

$$M = \begin{bmatrix} m(p_1) \\ m(p_2) \\ \vdots \\ m(p_n) \end{bmatrix} \tag{2.1}$$

In equation 2.1, $m(p_i)$ represents the number of tokens assigned to place p_i. An example of a marked Petri net (Petri net graph with tokens) is shown in Fig. 2.2

From the Fig 2.2, we can derive the marking M as:

$$M = \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} \tag{2.2}$$

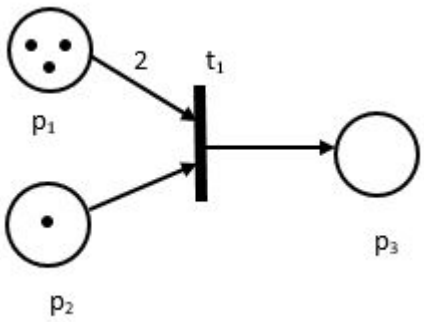

Fig. 2.2. An Example of a marked Petri Net

2.3 Petri net dynamics

Dynamics of a Petri net depends on firing of its enabled transitions. A transition t_j in a marked Petri net is said to be enabled when the number of tokens in each input place of t_j has at least as many tokens as the weight of the arc connecting that place with t_j. Mathematically speaking, a transition t_j in a Petri net is said to be enabled if,

$$m(p_i) \geq W(p_i, t_j) \quad \forall p_i \in I(t_j) \tag{2.3}$$

Consider an example of a marked Petri net as shown in Fig. 2.2. The marking M of this Petri net can be given as in Eq. 2.2. Here, we can say that transition t_1 is enabled since the number of tokens in each input place ($m(p_1) = 3$ and $m(p_2) = 1$) of transition t_1 is greater than or equal to the respective weights of the arc ($W(p_1, t_1) = 2$ and $W(p_2, t_1) = 1$) connecting them which satisfies the inequality 2.3.

If the Petri net marking shown in Fig. 2.2 is changed to a new marking M^1

$$M^1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

as shown in Fig. 2.3, transition t_1 cannot be enabled since $m(p_1) \ngeq W(p_1, t_1)$.

An enabled transition may fire, progressing the Petri net from one marked state to another by moving tokens through the net. The movement of tokens can be explained

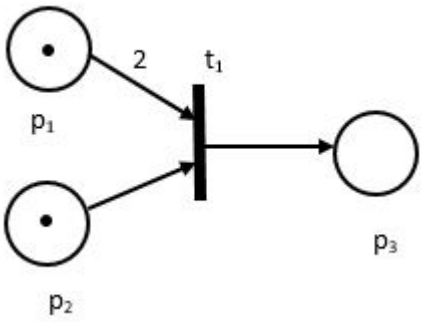

Fig. 2.3. Modified marked Petri net

as follows: when an enabled transition t_j fires, it removes as many tokens as the weight of the arc from each of its input place ($I(t_j)$); and deposits as many tokens as the weight of the arc to each of its output place ($O(t_j)$). Note that at each instant only one enabled transition may fire.

Consider the marked Petri net shown in Fig. 2.2 with its initial marking M given as in Eq. 2.2. In this example, we can say that transition t_1 is enabled (as discussed earlier in this section) and so it may fire. When t_1 fires, it first removes as many tokens as the weight of the arc from each of its input place ($I(t_1) = \{p_1, p_2\}$) and as a result two tokens from p_1 and one token from p_2 are removed making $m(p_1) = 1$ and $m(p_2) = 0$. Next, t_1 deposits as many tokens as the weight of the arc to each of its output place ($O(t_1) = \{p_3\}$) and as a result one token is deposited in place p_3 making $m(p_3) = 1$. The new state of Petri net after the firing of transition t_1 is shown in Fig. 2.4 and the marking of the Petri net changes to M_1 which is given as,

$$M_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \tag{2.4}$$

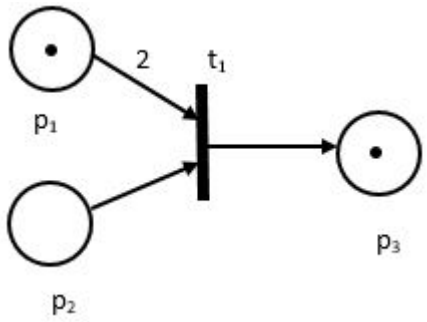

Fig. 2.4. New state of Petri net after t_1 is fired

2.4 Analysis of a Petri net

In this section, some methods to analyze a system modeled using Petri nets are discussed.

2.4.1 Incidence matrices

To study the dynamic behavior of Petri nets algebraically, matrix equations are implemented. Consider a Petri net with m places and n transitions, we can define:

Output Incidence Matrix: B^+ as a $(m \times n)$ matrix that captures all arc weights from T to P. That is,

$$B^+ = \begin{bmatrix} W(t_1, p_1) & W(t_2, p_1) & \cdots & W(t_n, p_1) \\ W(t_1, p_2) & W(t_2, p_2) & \cdots & W(t_n, p_2) \\ \vdots & \vdots & \ddots & \vdots \\ W(t_1, p_n) & W(t_2, p_n) & \cdots & W(t_n, p_n) \end{bmatrix}$$

The value of $W(t_j, p_i)$ is zero when no arc exists from t_j to p_i.

Input Incidence Matrix: B^- as a $(m \times n)$ matrix that captures all arc weights from P to T. That is,

$$B^- = \begin{bmatrix} W(p_1,t_1) & W(p_1,t_2) & \cdots & W(p_1,t_n) \\ W(p_2,t_1) & W(p_2,t_2) & \cdots & W(p_2,t_n) \\ \vdots & \vdots & \ddots & \vdots \\ W(p_n,t_1) & W(p_n,t_2) & \cdots & W(p_n,t_n) \end{bmatrix}$$

The value of $W(p_i,t_j)$ is zero when no arc exists from p_i to t_j.

Incident matrix: B as a $m \times n$ matrix obtained by the result of subtracting B^- from B^+. Mathematically showing,

$$B = B^+ - B^- \tag{2.5}$$

Consider the Petri net shown in Fig. 2.1 with three places and one transition. The output incidence matrix B^+ and input incidence martix B^- have dimensions 3×1 and can be given as follows:

$$B^+ = \begin{bmatrix} W(t_1,p_1) \\ W(t_1,p_2) \\ W(t_1,p_3) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

$$B^- = \begin{bmatrix} W(p_1,t_1) \\ W(p_2,t_1) \\ W(p_3,t_1) \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix} \tag{2.6}$$

2.4.2 State equation of a Petri net

The state equation of a Petri net gives an algebraic tool alternative to graphical way of describing the firing of transitions and changing the state of a Petri net.

Given, a current state of Petri net with marking M_k and the case that a particular transition t_j is fired, the next state of the Petri net with marking M_{k+1} can be obtained using the state equation of a Petri net (Eq. 2.7).

$$M_{k+1} = M_k + BX_k \tag{2.7}$$

Here,

M_{k+1} is the marking at time instant $k+1$

M_k is the marking at time instant k

B is the incidence matrix

X_k is the firing vector at time instant k. It is a column vector with dimensions $m \times 1$ where m is the number of transitions in a Petri net. It has only one non-zero entry with a value one indicating which transition fires. The form of a firing vector is shown in the Eq. 2.8 where one appears only in j^{th} row ($j \in \{1, \cdots, m\}$), indicating the fact that transition t_j is currently firing.

$$X_k = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.8}$$

Consider the Petri net example shown in Fig. 2.1 with its initial marking M_0 given in Eq. 2.2. By using the state equation, let us now mathematically calculate the next state of the Petri net (with marking M_1) after the enabled transition t_1 fires. The firing vector X_0 defines that the transition t_1 is being fired. The firing vector will

be a 1×1 matrix as the Petri net has only one transition t_1. The incidence matrix B for this Petri net is already discussed and calculated as shown in Eq. 2.6.

$$\begin{aligned} M_1 &= M_0 + BX_0 \\ &= \begin{bmatrix} 3 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 \\ -1 \\ 1 \end{bmatrix} \begin{bmatrix} 1 \end{bmatrix} \\ &= \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \end{aligned}$$

We can observe that the new marking M_1 obtained algebraically using state equation is identical to the marking obtained graphically as shown in Eq. 2.4.

2.4.3 Place Invariant

Place invariant (P-invariant) is one of the structural properties of a Petri net which yields information about the token conservation in a Petri net. A vector X is a place invariant if it satisfies,

$$X^T B = 0 \tag{2.9}$$

Here,

X is a $m \times 1$ column vector where m is the number of places in a Petri net.

B is the incidence matrix.

Consider the state equation of a Petri net as discussed in Section 2.4.2,

$$M_{k+1} = M_0 + BV_k$$

Here,

M_0 is the initial marking.

V_k is the firing vector at time instant k.

Left multiplying the state equation with X^T we get,

$$X^T M_{k+1} = X^T M_0 + X^T B V_k \tag{2.10}$$

Since $X^T B = 0$ from Eqn. 2.9, substituting it in Eqn. 2.10 we get,

$$X^T M_{k+1} = X^T M_0 \tag{2.11}$$

From the Eqn .2.11, it is clear that for any chosen M_0, the number of tokens in a set of places weighted by X is a constant for all markings M_{k+1} that are reachable from M_0. This set of places is said to be covered by a P-invariant. If every place in a Petri net is covered by a P-invariant, we say that the Petri net is bounded. This analysis shows the conservation properties of a Petri net.

2.4.4 Transition Invariant

Transition invariant (T-invariant) is another structural property of a Petri net which yields information about loops present in a Petri net. Consider the state equation of a Petri net as discussed in Section 2.4.2.

$$M_{k+1} = M_0 + B V_k$$

Here,

M_0 is the initial marking.

V_k is the firing vector at time instant k.

B is the incidence matrix.

If there exists a vector Y (of order $n \times 1$ where, n represents the number of transitions in a Petri net) such that:

$$BY = 0 \tag{2.12}$$

and there exists a valid transition firing sequence corresponding to V_k, then Y is called the transition invariant of the Petri net. A T-invariant can lead the marking (state) back to the same marking (state) after firing a sequence of transitions. The entries of a T-invariant define the firing counts of the respective transitions that are present in the firing sequence. However, the T-invariant cannot determine the order of the transitions in the firing sequence.

2.4.5 Reachability

Reachability is one of the important behavioral properties of a Petri net. To see if the system modeled using Petri net meets the requirements by reaching all the desired states and by not reaching undesired states is an important problem. Reachability helps to determine whether the Petri net model is able to reach a particular state M from an initial state M_0. A sequence of transition firings occur when the state M_0 is transformed to a state M and this sequence of firings show a functional behavior of the system. There can be multiple sequences of transition firings which can transform the state M_0 to state M leading to possibility of different or unanticipated functional behavior of the system reflected with the help of Petri net model.

A Petri net marking M^1 is said to be reachable from another marking M, if there exists a firing sequence of transitions that transform marking M to marking M^1. Reachable set of a Petri net is defined as the set of all markings that are reachable from the initial marking M_0 of the Petri net.

Given an initial marking M_0 of the Petri net, we can obtain as many new markings (M_i) as the number of enabled transitions (t_j) which satisfy the inequality 2.13.

$$M_0 \geq B^-(:, t_j) \tag{2.13}$$

When an enabled transition t_j fires, the respective new marking M_i can be obtained by using the state equation of the Petri net as shown in Eq. 2.14.

$$M_i = M_0 + BX_0 \tag{2.14}$$

From each of these new markings, we can further reach more markings. This dynamics can be represented in the form of a tree called coverability tree. For a bounded Petri net, it can be also named as reachability tree.

Nodes of this tree represent initial marking and markings reachable from it.

Arcs represent the fired transitions which transform a marking to another.

2.5 Time Petri nets

Time Petri net is a model introduced by Merlin and Farber to overcome the limitations of ordinary Petri nets to include timing constraints [14]. Time Petri nets add the variable, time, into ordinary Petri nets. In these type of Petri nets, two time values (α and β) are introduced for each transition.

α represents minimum time waited by an enabled transition to fire.

β represents maximum time waited by an enabled transition to fire.

In other words, an enabled transition fires with delay $\theta \in [\alpha, \beta]$. The default value of α is zero and β is infinite. Merlin defined these time delay intervals for each transition t_j to fire as static rational values and termed the time interval as static firing interval of transition t_j. The lower bound (α) is termed as the static earliest firing time (static EFT) and the upper bound (β) is termed as the static latest firing time (static LFT).

Time Petri nets are used in applications such as modeling and verification of real-time concurrent systems and scheduling jobs [15–18].

2.6 Simulation tools

Simulation is important for understanding the behavior of a system. With the same idea, Petri net models are simulated to analyze the properties and performance

and to gain more insight into the model. Ideally, simulation of a Petri net incorporates an algorithm to run the Petri net [19]. Such an algorithm is designed as part of this thesis and a MATLAB code is developed implementing the algorithm. In addition, software tool named Netlab [20] is used for simulation and to validate the results obtained from the MATLAB code developed. Another software tool named Tina [21] is used to simulate the developed Time Petri net.

3. MATHEMATICAL MODELING

In this chapter, the design of a Petri net model for Muzima fingerprint module is discussed. To model any system using Petri nets, detail understanding of the system's workflow is necessary. The following steps are implemented as a design process to model Muzima fingerprint module using Petri nets:

1. Construct a detailed workflow of Muzima fingerprint module using a flow chart.
2. Identify various subflows embedded in the flow chart.
3. Design a Petri net for each subflow identified.
4. Combine all the individual Petri nets to form the final Petri net model for Muzima fingerprint module.
5. Identify the initial marking (initial state of the Petri net) and add to the Petri net model developed.

In this design process,

Places represent conditions, input/output data.

Transitions represent tasks, clauses.

3.1 Workflow of Muzima fingerprint module

The workflow of Muzima fingerprint module was discussed briefly in Section 1.2. Following is a detailed explanation of the workflow along with graphical representation using flow chart as shown in Fig. 3.1. The circles A, B and C shown in the Fig. 3.1 represent connectors in the flow chart.

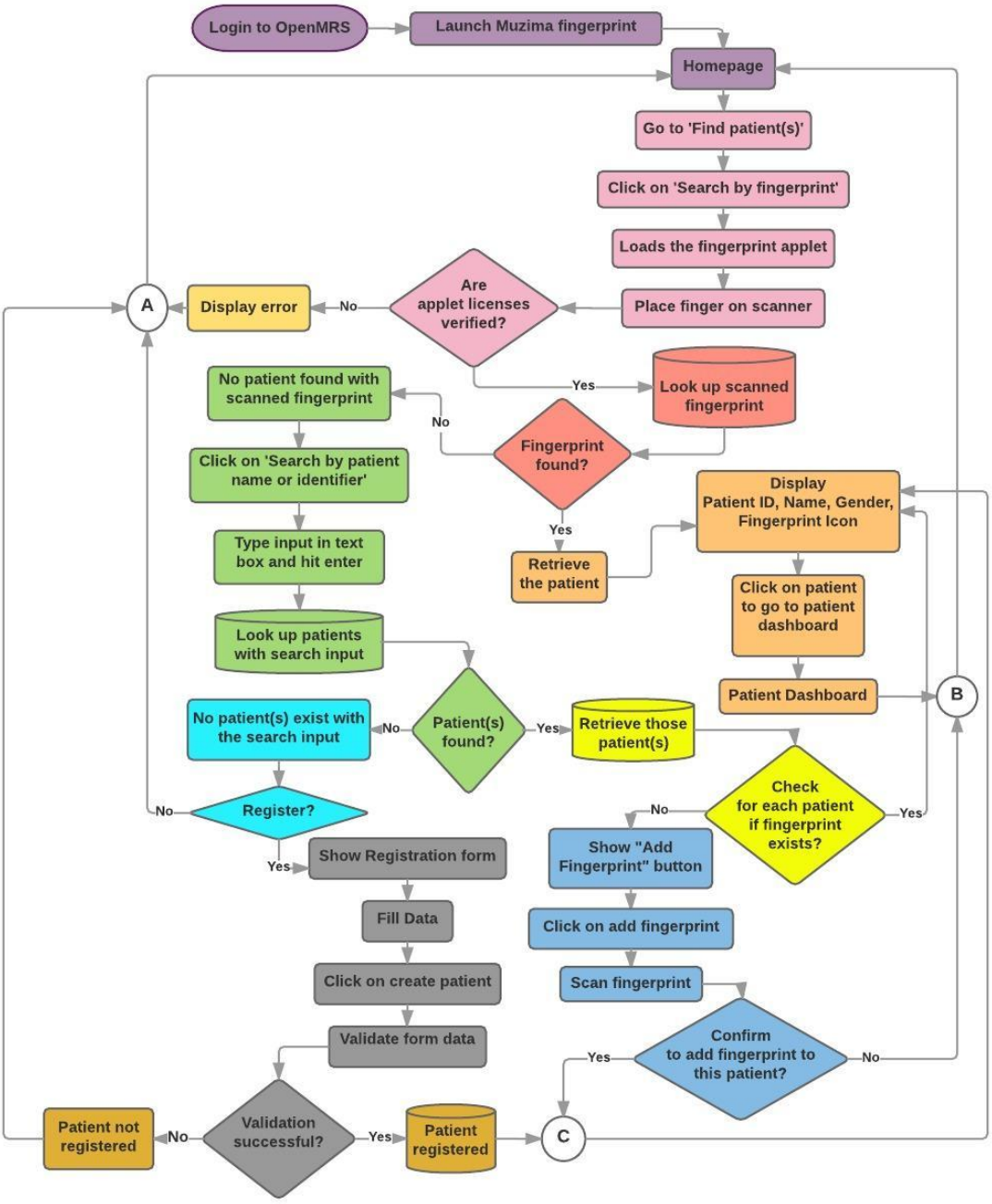

Fig. 3.1. Detailed workflow of Muzima fingerprint module

1. Log in to OpenMRS and click on the Muzima fingerprint module shown under the modules list. This will launch the Muzima fingerprint and displays its homepage.

2. Go to 'Find patient(s)' section and click on 'Search by fingerprint' to find patient using patient's fingerprint. This will load the fingerprint applet. Once it loads, place finger on the scanner. If the fingerprint applet licenses,

   (a) are verified successfully, the scanning process will complete without errors. After scanning completes, the module looks up the EMR database of fingerprints of registered patients. If the fingerprint,

      i. is identified, the module retrieves the associated patient record, displays patient details such as patient ID, first name, last name and gender.

      ii. is not identified, a message 'No patient found with scanned fingerprint' is displayed to the user. It is either because the patient is not registered in the EMR or the patient record is not updated with the fingerprint data. In this case, we can search by patient name or identifier. Type the search input in the text box and hit enter. The module looks up the EMR database for the search input. If,

         A. patient(s) found, the module retrieves all those patient(s) records who match with the search input. The module displays the patient details for each retrieved patient and gives an option to add fingerprint for those patients whose fingerprint data is missing. If chosen to add fingerprint to the patient, click on the 'Add fingerprint' button. Then scan the finger and confirm to add fingerprint to the patient when prompted.

         B. no patient(s) found, an option is given to register the patient into the EMR database. If chosen to register, the module brings up the registration form. After filling the form, click on 'Create pa-

tient' button. Then the registration form data is validated and the patient is registered if the validation was successful and otherwise not.

(b) are not verified, scanning stops and the module returns to homepage displaying an error message to the user.

3. In any of the above situations where the module displays the patient record, clicking on the patient record opens up the patient dashboard for more details and the user can go back to the homepage.

3.2 Modularization

In this section, various sub-flows of the flow chart shown in Fig. 3.1 are identified and organized into modules with a petri net modeled for each of them.

- Sub-flow 1: Login to OpenMRS system and launch Muzima fingerprint module. This will bring up the homepage for Muzima fingerprint module. Fig. 3.2 represents the flow chart and Fig. 3.3 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.1.

Fig. 3.2. Flow chart representation of Sub-flow 1

- Sub-flow 2: From the homepage of Muzima fingerprint module, go to 'Find patient(s)' section. Click on 'Search by fingerprint' button to search patient by patient's fingerprint. This will load the fingerprint applet. Once loaded, place finger on scanner. If the applet licenses are,

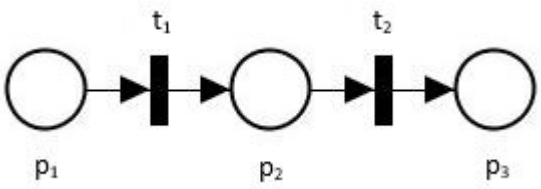

Fig. 3.3. Petri net representation of Sub-flow 1

Table 3.1 Places and transitions of Petri net model for Sub-flow 1

Place	**Description**	**Transition**	**Label**
p_1	Start	t_1	Login to OpenMRS
p_2	Logged in	t_2	Launch Muzima fingerprint module
p_3	Homepage		

  - verified successfully, scanning completes and the module looks up for the scanned fingerprint in EMR database of fingerprints of registered patients.
  - not verified, the module returns to homepage with an error message to the user.

  Fig. 3.4 represents the flow chart and Fig. 3.5 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.2.

- Sub-flow 3: The Muzima fingerprint module looks up for the scanned fingerprint in the EMR database. If the fingerprint is,

  - identified, the module retrieves the patient from the EMR database and displays the patient record with basic information such as id, first name, last name and gender. By clicking on the patient record, the module navigates to the patient dashboard where the complete patient details can be viewed and can return back to homepage.

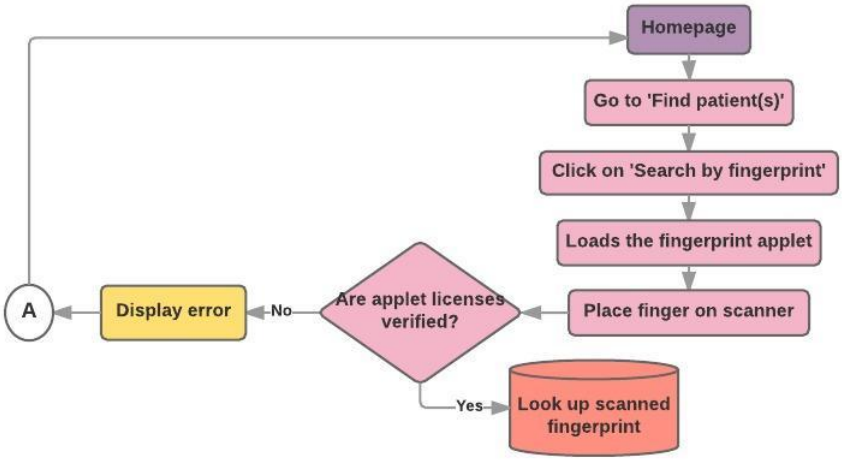

Fig. 3.4. Flow chart representation of Sub-flow 2

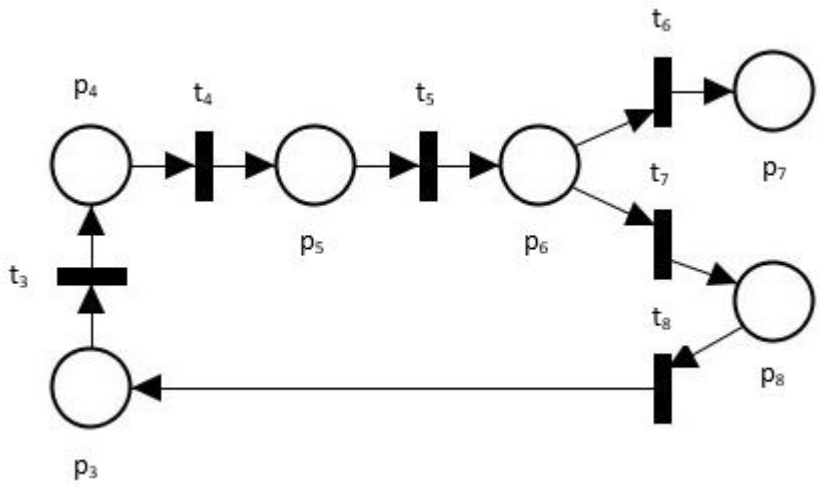

Fig. 3.5. Petri net representation of Sub-flow 2

– not identified, a message 'No patient found with the scanned fingerprint' is displayed to the user.

Fig. 3.6 represents the flow chart and Fig. 3.7 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.3.

Table 3.2 Places and transitions of Petri net model for Sub-flow 2

Place	**Description**	**Transition**	**Label**
p_3	Homepage	t_3	Go to 'Find patient(s)' section
p_4	Find patient(s) section	t_4	Click on 'Search by fingerprint'
p_5	Fingerprint applet loaded	t_5	Place finger on the scanner
p_6	Check if fingerprint applet licenses are verified	t_6	Licenses are verified successfully
p_7	System triggered to look up for scanned fingerprint	t_7	Licenses are not verified
p_8	Error displayed	t_7	Licenses are not verified
		t_8	Go back to Homepage

- Sub-flow 4: If no patient is found with the scanned fingerprint, it is either because the patient is not registered in the EMR or the patient record is not updated with the fingerprint data. In this case, we can search by patient name or identifier. Click on 'Search by patient name or identifier'. Type input data in text box and hit enter. The module looks up for patient(s) whose name or identifier match with the search input. If patient(s)
  - found, the module retrieves all those patients.
  - not found, a message 'No patient(s) exist with the search input' is displayed.

  Fig. 3.8 represents the flow chart and Fig. 3.9 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.4.

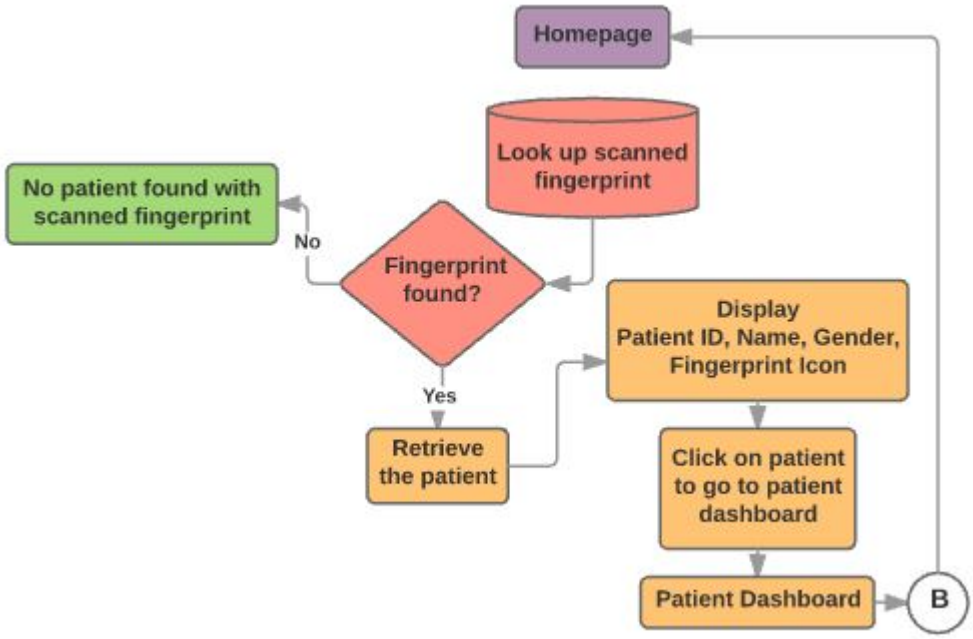

Fig. 3.6. Flow chart representation of Sub-flow 3

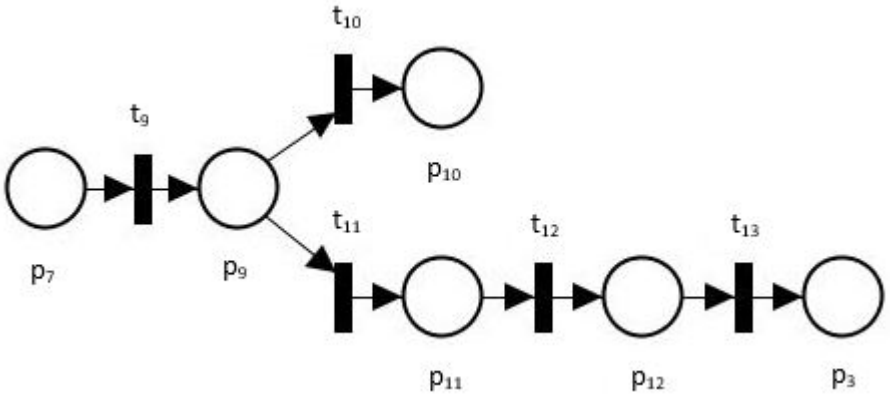

Fig. 3.7. Petri net representation of Sub-flow 3

- Sub-flow 5: If no patient(s) are found with the search input, that implies the patient is not registered in the EMR system. So the module gives an option to register the patient as a new patient. If,

Table 3.3 Places and transitions of Petri net model for Sub-flow 3

Place	**Description**	**Transition**	**Label**
p_3	Homepage	t_9	Search all fingerprint data
p_7	System triggered to look up for scanned fingerprint	t_{10}	Fingerprint not identified
p_9	Check if fingerprint found	t_{11}	Fingerprint identified
p_{10}	Message displayed as 'No patient found with the scanned fingerprint'	t_{12}	Click on patient record to go to patient dashboard
p_{11}	Patient record with id, first name, last name and gender is displayed	t_{13}	Return to homepage
p_{12}	Patient dashboard		

- chosen to register, the module brings up the registration form. After filling the form, click on 'Create patient' button to submit the form. Then the form data is validated. If the validation is,
  - successful, patient is registered into the EMR system. The module then displays the newly registered patient record with basic information such as id, first name, last name and gender.
  - not successful, patient registration fails and and one can return to homepage.
- not chosen to register, the module returns to the homepage.

Fig. 3.10 represents the flow chart and Fig. 3.11 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.5.

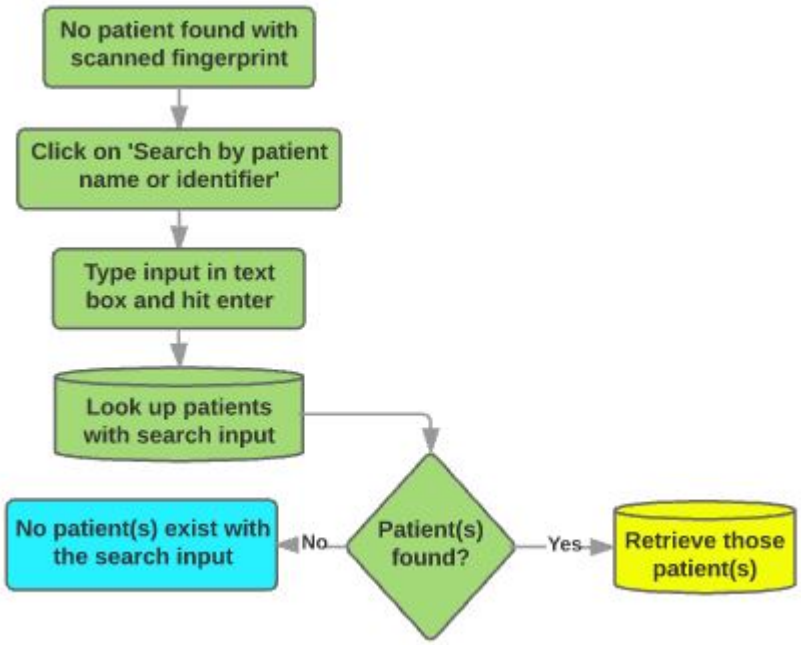

Fig. 3.8. Flow chart representation of Sub-flow 4

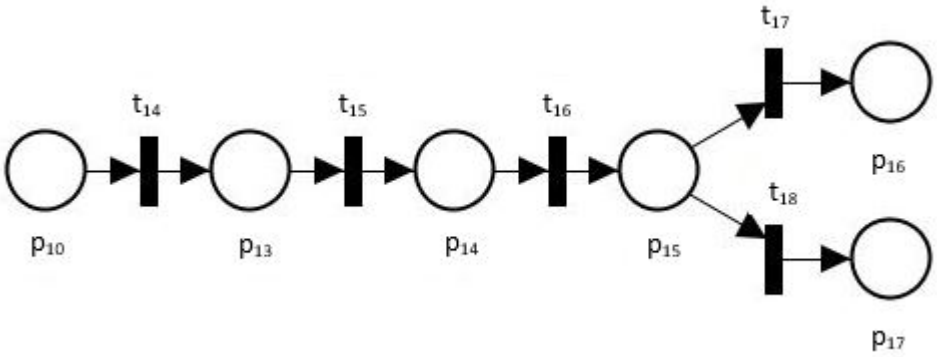

Fig. 3.9. Petri net representation of Sub-flow 4

- Sub-flow 6: When patient(s) are found using search by patient name or identifier, the module retrieves those patients. Then, it checks for each whether the patient records are updated with fingerprint information. If the retrieved patient record,
  - contains fingerprint data, the patient is displayed to the user.

Table 3.4 Places and transitions of Petri net model for Sub-flow 4

Place	**Description**	**Transition**	**Label**
p_{10}	Message displayed as 'No patient found with the scanned fingerprint'	t_{14}	Click on 'Search by patient name or identifier'
p_{13}	Search box shown	t_{15}	Type name or identifier in search box and hit enter
p_{14}	System triggered to look up for search input	t_{16}	Search all patient names and identifiers
p_{15}	Check if patient(s) found	t_{17}	Patient(s) found
p_{16}	System triggered to retrieve all those patient(s)	t_{18}	Patient(s) not found
p_{17}	Message displayed as 'No patient(s) exist with the search input'		

– does not contain fingerprint data, an 'Add fingerprint' button is shown with respect to the patient. Click on the button to scan the patient finger. After scanning completes, confirm to add fingerprint to the patient record. If,

  * confirmed to add scanned fingerprint, the patient record is updated with fingerprint and the module displays the patient.
  * not confirmed to add scanned fingerprint, the module returns to home-page.

Fig. 3.12 represents the flow chart and Fig. 3.13 represents the Petri net model for this sub-flow. The descriptions of the places and transitions of this Petri net model are given in Table. 3.6.

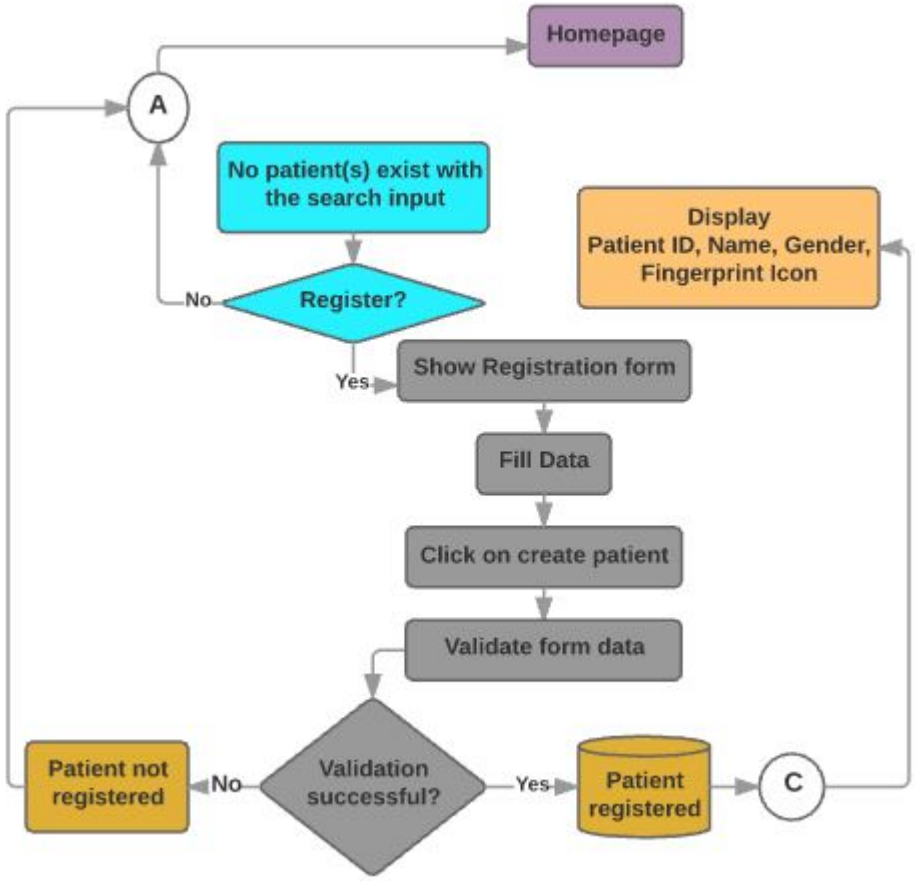

Fig. 3.10. Flow chart representation of Sub-flow 5

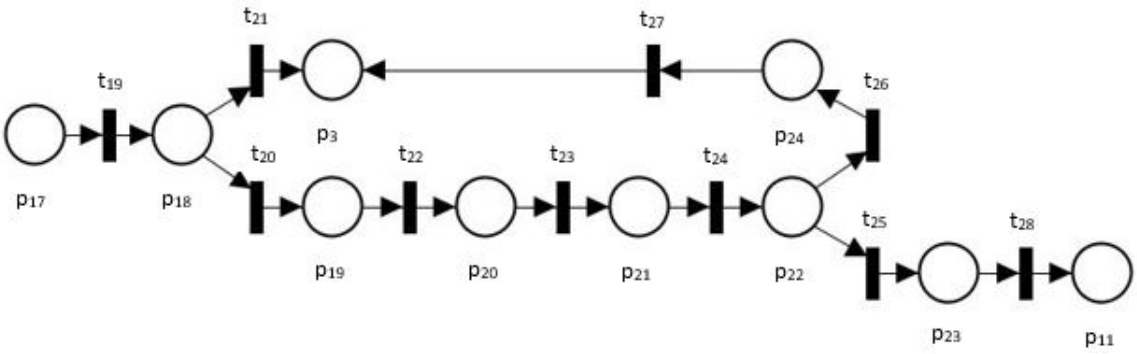

Fig. 3.11. Petri net representation of Sub-flow 5

Table 3.5 Places and transitions of Petri net model for Sub-flow 5

Place	**Description**	**Transition**	**Label**
p_3	Homepage	t_{19}	Look for available actions
p_{11}	Patient record with id, first name, last name and gender is displayed	t_{20}	Choose to register
p_{17}	Message displayed as 'No patient(s) exist with the search input'	t_{21}	Choose to not register
p_{18}	Provide option to register	t_{22}	Fill in form data
p_{19}	Registration form displayed	t_{23}	Click on 'Create patient'
p_{20}	Filled out registration form	t_{24}	Validate form data
p_{21}	Form ready for validation	t_{25}	Validation successful
p_{22}	Check validation results	t_{26}	Validation unsuccessful
p_{23}	Patient registered	t_{27}	Return to homepage
p_{24}	Patient not registered	t_{28}	Get patient data to display

3.3 Ordinary Petri net model for Muzima fingerprint module

In Section 3.2, Petri net models were developed for each of the sub-flow of Muzima fingerprint module. In this section, all these individual Petri nets are connected to form the complete Petri net model. Fig. 3.14 shows the complete Petri net model for Muzima fingerprint module. This Petri net model consists a total of 28 places and 35 transitions. The descriptions of the places and transitions of the complete Petri net model are given in Table. 3.7.

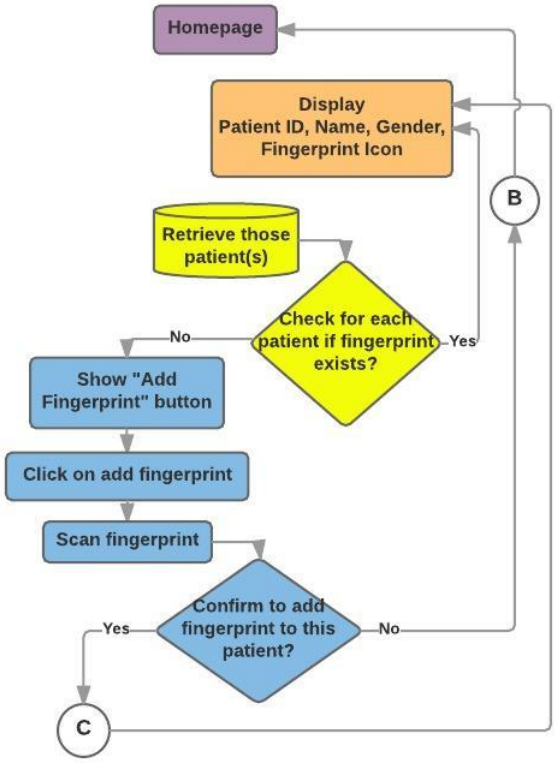

Fig. 3.12. Flow chart representation of Sub-flow 6

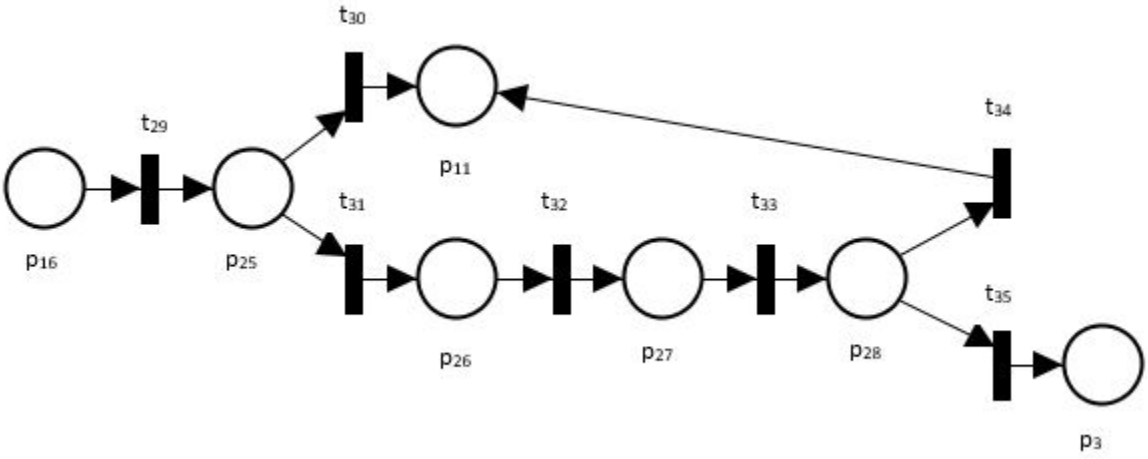

Fig. 3.13. Petri net representation of Sub-flow 6

Table 3.6 Places and transitions of Petri net model for Sub-flow 6

Place	**Description**	**Transition**	**Label**
p_3	Homepage	t_{29}	Retrieve the patient records
p_{11}	Patient record with id, first name, last name and gender is displayed	t_{30}	Fingerprint exists
p_{16}	System triggered to retrieve all those patient(s)	t_{31}	Fingerprint does not exists
p_{25}	Check for each patient if fingerprint exists	t_{32}	Click on 'Add fingerprint' button
p_{26}	'Add fingerprint' button shown	t_{33}	Scan fingerprint
p_{27}	System ready to start scanning process	t_{34}	Confirm to add
p_{28}	Check if confirm to add fingerprint to patient	t_{35}	Do not confirm to add

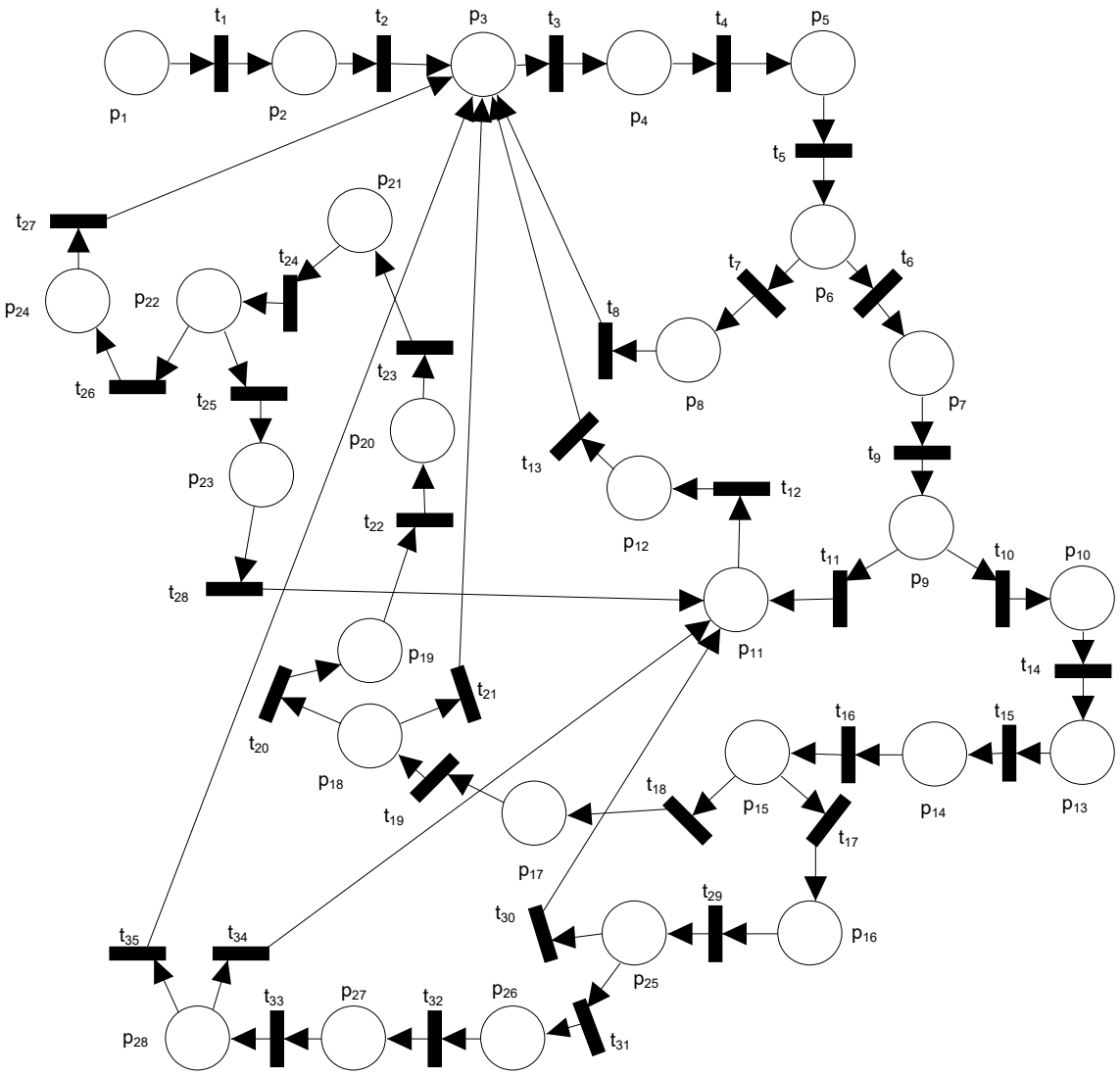

Fig. 3.14. Petri net model for Muzima fingerprint module

Table 3.7: Places and transitions of the complete Petri net model

Place	**Description**	**Transition**	**Label**
p_1	Start	t_1	Login to OpenMRS
p_2	Logged in	t_2	Launch Muzima fingerprint module
p_3	Homepage	t_3	Go to 'Find patient(s)' section
p_4	Find patient(s) section	t_4	Click on 'Search by fingerprint'
p_5	Fingerprint applet loaded	t_5	Place finger on the scanner
p_6	Check if fingerprint applet licenses are verified	t_6	Licenses are verified successfully
p_7	System triggered to look up for scanned fingerprint	t_7	Licenses are not verified
p_8	Error displayed	t_8	Go back to Homepage
p_9	Check if fingerprint found	t_9	Search all fingerprint data
p_{10}	Message displayed as 'No patient found with the scanned fingerprint'	t_{10}	Fingerprint not identified
p_{11}	Patient record with id, first name, last name and gender is displayed	t_{11}	Fingerprint identified
p_{12}	Patient dashboard	t_{12}	Click on patient record to go to patient dashboard
p_{13}	Search box shown	t_{13}	Return to homepage
p_{14}	System triggered to look up for search input	t_{14}	Click on 'Search by patient name or identifier'

continued on next page

Table 3.7: *continued*

Place	**Description**	**Transition**	**Label**
p_{15}	Check if patient(s) found	t_{15}	Type name or identifier in search box and hit enter
p_{16}	System triggered to retrieve all those patient(s)	t_{16}	Search all patient names and identifiers
p_{17}	Message displayed as 'No patient(s) exist with the search input'	t_{17}	Patient(s) found
p_{18}	Provide option to register	t_{18}	Patient(s) not found
p_{19}	Registration form displayed	t_{19}	Look for available actions
p_{20}	Filled out registration form	t_{20}	Choose to register
p_{21}	Form ready for validation	t_{21}	Choose to not register
p_{22}	Check validation results	t_{22}	Fill in form data
p_{23}	Patient registered	t_{23}	Click on 'Create patient'
p_{24}	Patient not registered	t_{24}	Validate form data
p_{25}	Check for each patient if fingerprint exists	t_{25}	Validation successful
p_{26}	'Add fingerprint' button shown	t_{26}	Validation unsuccessful
p_{27}	System ready to start scanning process	t_{27}	Return to homepage
p_{28}	Check if confirm to add fingerprint to patient	t_{28}	Get patient data to display
		t_{29}	Retrieve the patient records
		t_{30}	Fingerprint exists
		t_{31}	Fingerprint does not exists

continued on next page

Table 3.7: *continued*

Place	**Description**	**Transition**	**Label**
		t_{32}	Click on 'Add fingerprint' button
		t_{33}	Scan fingerprint
		t_{34}	Confirm to add
		t_{35}	Do not confirm to add

3.4 Initial marking

In this section, token(s) are assigned to the Petri net model (Fig. 3.14) developed for Muzima fingerprint module which defines the initial marking for this model. A Petri net marking is defined as the assignment of token(s) to places in a Petri net. The dynamics of a Petri net may lead to change in the position as well as number of tokens [19]. The initial assignment of token(s) provides the initial marking M_0 of the Petri net. To execute the Petri net model developed, a token is assigned to the place p_1 indicating the start of the process. Fig. 3.15 shows the marked Petri net for Muzima fingerprint module. The initial marking M_0 is given as a 28×1 matrix with a non-zero value one in the first row representing the token in place p_1 i.e., $m(p_1) = 1$.

$$M_{0(28\times1)} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \quad (3.1)$$

3.5 Time Petri net model for Muzima fingerprint module

As discussed in Section. 2.5, we can introduce timing constraints to transitions in a Petri net which are expressed as the minimum and maximum time values taken by an enabled transition to fire. Fig. 3.16 shows the Time Petri net model developed for Muzima fingerprint module. The time delays of all the transitions are given in Table. 3.8.

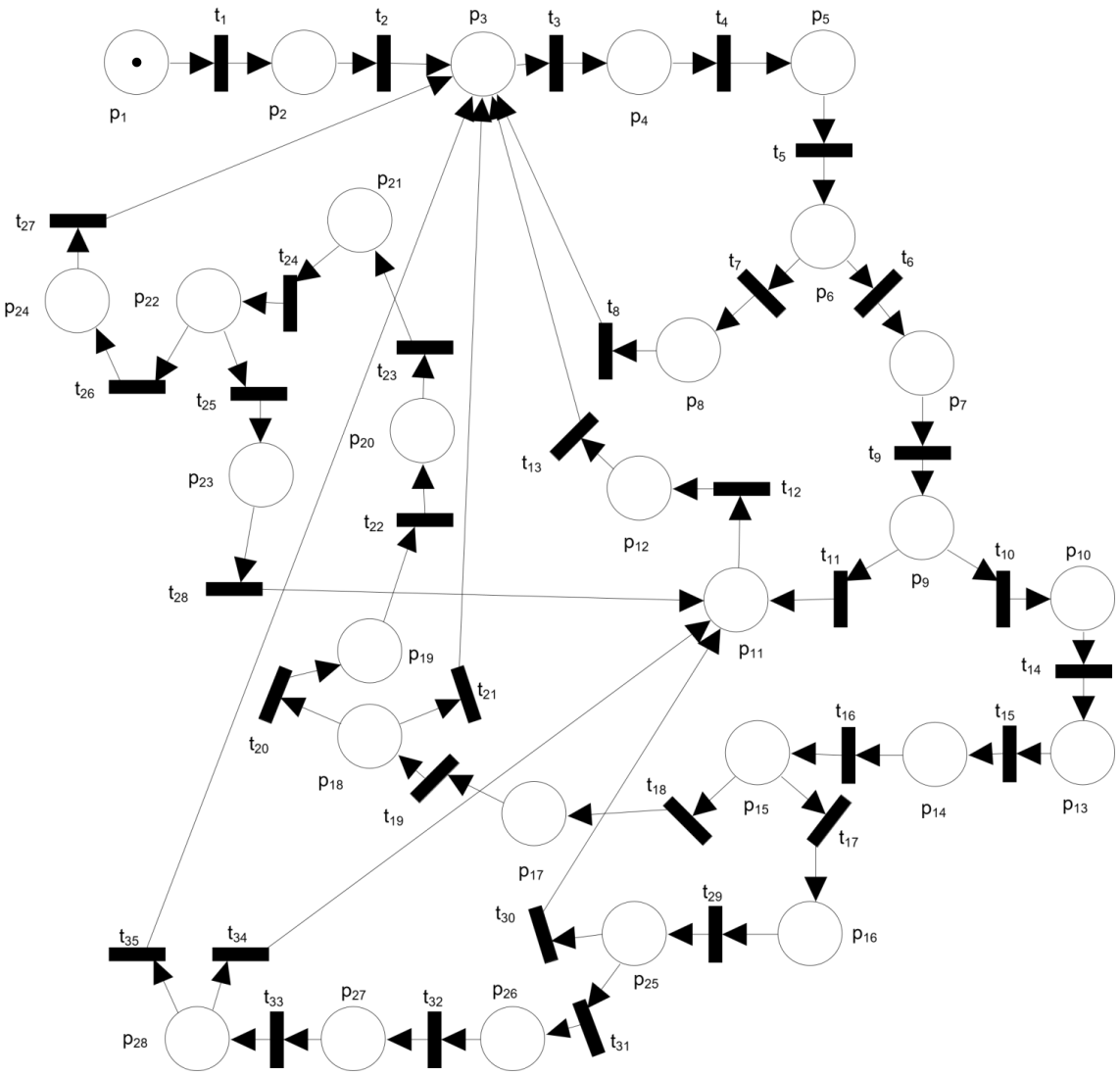

Fig. 3.15. Petri net with initial marking

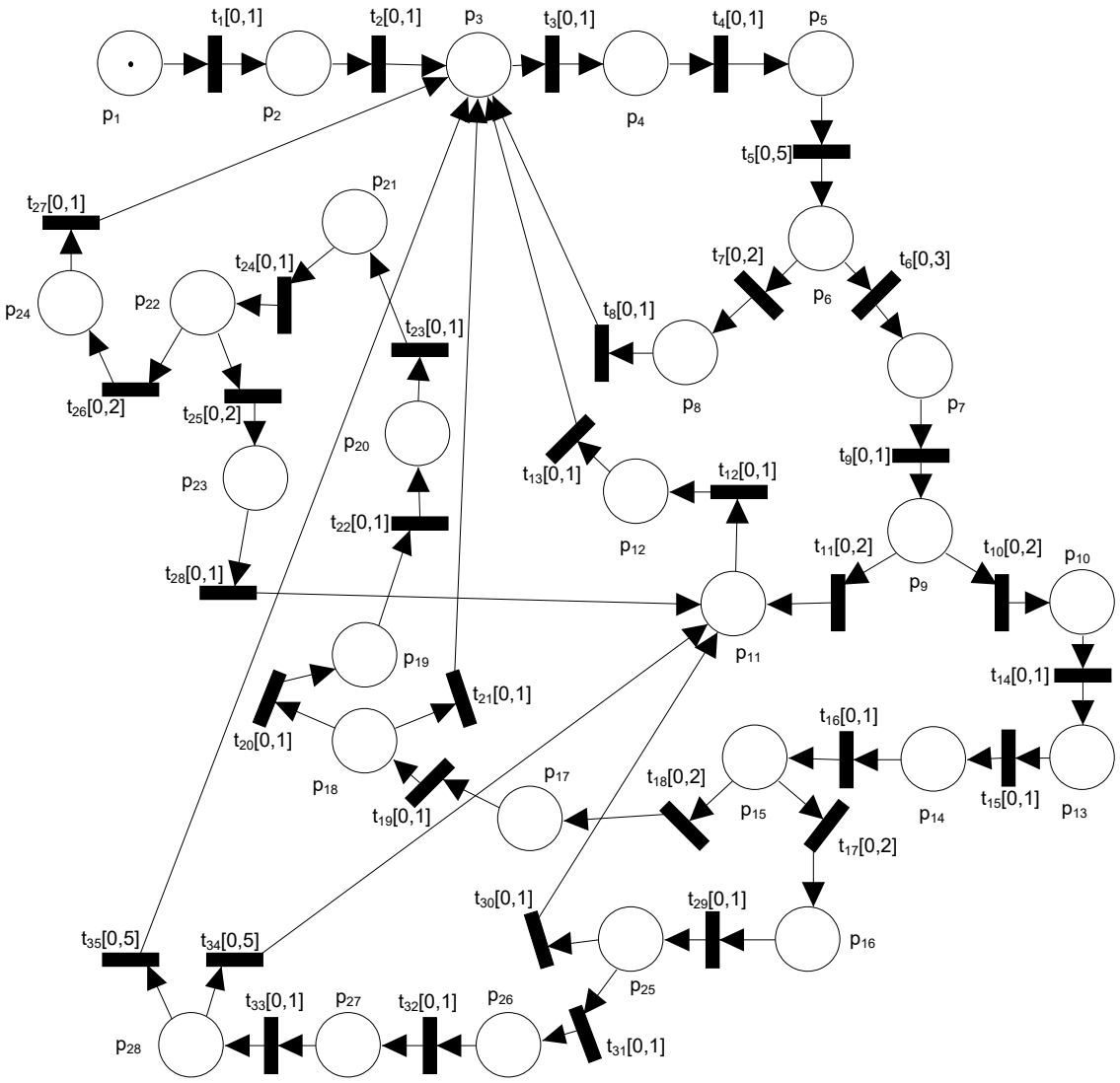

Fig. 3.16. Time Petri net model for Muzima fingerprint module

Table 3.8: Transitions and their time delays

Transition	**Description**	**Time delay**
t_1	Login to OpenMRS	[0,1]
t_2	Launch Muzima fingerprint module	[0,1]
t_3	Go to 'Find patient(s)' section	[0,1]
t_4	Click on 'Search by fingerprint'	[0,1]
t_5	Place finger on the scanner	[0,5]
t_6	Licenses are verified successfully	[0,3]
t_7	Licenses are not verified	[0,2]
t_8	Go back to Homepage	[0,1]
t_9	Search all fingerprint data	[0,1]
t_{10}	Fingerprint not identified	[0,2]
t_{11}	Fingerprint identified	[0,2]
t_{12}	Click on patient record to go to patient dashboard	[0,1]
t_{13}	Return to homepage	[0,1]
t_{14}	Click on 'Search by patient name or identifier'	[0,1]
t_{15}	Type name or identifier in search box and hit enter	[0,1]
t_{16}	Search all patient names and identifiers	[0,1]
t_{17}	Patient(s) found	[0,2]
t_{18}	Patient(s) not found	[0,2]
t_{19}	Look for available actions	[0,1]
t_{20}	Choose to register	[0,1]
t_{21}	Choose to not register	[0,1]
t_{22}	Fill in form data	[0,1]
t_{23}	Click on 'Create patient'	[0,1]
t_{24}	Validate form data	[0,1]

continued on next page

Table 3.8: *continued*

Transition	**Description**	**Time delay**
t_{25}	Validation successful	[0,2]
t_{26}	Validation unsuccessful	[0,2]
t_{27}	Return to homepage	[0,1]
t_{28}	Get patient data to display	[0,1]
t_{29}	Retrieve the patient records	[0,1]
t_{30}	Fingerprint exists	[0,1]
t_{31}	Fingerprint does not exists	[0,1]
t_{32}	Click on 'Add fingerprint' button	[0,1]
t_{33}	Scan fingerprint	[0,1]
t_{34}	Confirm to add	[0,5]
t_{35}	Do not confirm to add	[0,5]

4. ANALYSIS

Having developed the mathematical model for the Muzima fingerprint module based on Petri nets in Section 3.3, there is a need to analyze this model to gain more insight about the behavior of the system. In this chapter, qualitative and quantitative analysis is performed on the Petri net model developed for the Muzima fingerprint module. To start the analysis, the incidence matrices of the Petri net model need to be obtained first.

4.1 Incidence matrix

The Petri net model developed for Muzima fingerprint module (Fig. 3.14) has 28 places and 35 transitions. Therefore, the dimensions of the output incidence matrix B^+, the input incidence matrix B^- and the incidence matrix B for this Petri net model are all 28×35.

The computation of the incidence matrices for a given Petri net is discussed in Section 2.4.1. Similarly, the output incidence matrix B^+ and input incidence matrix B^- for this Petri net model are calculated and shown in Eqn. 4.1 and Eqn. 4.2 respectively. The incidence matrix B is then calculated using B^+ and B^- and is shown in Eqn. 4.3.

$$
B^{+} = \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&0&1&0&0&0&0&1&0&0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&0&0&0&0&1\\
0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&1&0&0&0&1&0\\
0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0&1&0&0
\end{bmatrix} \tag{4.1}
$$

$$B^- = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1
\end{bmatrix} \tag{4.2}$$

$$B = \begin{bmatrix}
-1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & -1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{bmatrix} \tag{4.3}$$

4.2 Qualitative analysis

In qualitative analysis of the Petri net model, two structural properties are analyzed. These properties of the Petri net are place invariant (P-invariant) and transition invariant (T-invariant).

4.2.1 Place invariant

As discussed in Section 2.4.3, place invariants (P-invariants) are subsets of places of a Petri net, where the algebraic sum of tokens present in these places remains constant throughout the execution of the Petri net. P-invariants yield information about conservation and boundedness properties of a Petri net.

Place invariant analysis is performed on Muzima fingerprint module modeled using Petri net. P-invariant(s), if exist for this Petri net model, are obtained by solving the Eqn. 2.9 using the incidence matrix B shown in Eqn. 4.3.

$$X^T B = 0$$

where,

$$X_{28\times 1} = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_{26} & x_{27} & x_{28} \end{bmatrix}^T$$

x_i is the weight associated with the place p_i so that the weighted sum of the tokens of these places remains a constant for all markings that are reachable from initial marking. Solving for the P-invarints (X), we get:

$$x_1 = x_2 = x_3 = \cdots = x_{26} = x_{27} = x_{28}$$

Let,

$$x_1 = x_2 = x_3 = \cdots = x_{26} = x_{27} = x_{28} = 1$$

Then, the solution X can be given as:

$$X_{28\times 1} = \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots & 1 & 1 & 1 \end{bmatrix}^T$$

where all its entries are one. This indicates that all the places in the Petri net model are covered by the P-invariant. This implies that the sum of tokens in all the places is constant ($= 1$) for all states (markings M_k) that are reachable from the initial state (marking M_0 given in Eqn. 3.1) indicating token conservation. Since every place in the Muzima fingerprint Petri net model is covered by the P-invariant, the Petri net is said to be bounded.

4.2.2 Transition invariant

Transition invariant (T-invariant) analysis of a Petri net yields information about loops present in a Petri net. A T-invariant can lead the marking (state) back to the same marking (state) after firing a sequence of transitions. The entries of a T-invariant define the firing counts of the corresponding transitions that are present in the firing sequence.

T-invariant analysis is performed on the Muzima fingerprint module modeled using Petri nets to determine the functional loops present in the system. T-invariant(s), if exist for this Petri net model, are obtained by solving the Eqn. 2.12 using the incidence matrix B shown in Eqn. 4.3,

$$BY = 0$$

where,

$$Y_{35\times 1} = \begin{bmatrix} y_1 & y_2 & y_3 & \cdots & y_{33} & y_{34} & y_{35} \end{bmatrix}^T$$

y_i is the firing count of transition t_i that is present in the firing sequence. Solving the equation resulted in eight T-invariants for Muzima fingerprint Petri net model which are listed in Table. 4.1. For instance, one of the T-invariants firing sequence $t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_7 \rightarrow t_8$ corresponds to a process of performing patient search by scanning fingerprint from the homepage. Due to the fingerprint applet licenses not verified, the module is redirected back to homepage indicating a loop in the system.

Table 4.1: Summary of T-invariants in the Muzima fingerprint Petri net model

T-invariant	**Remark**
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_7 \rightarrow t_8$	Return back to homepage when fingerprint applet licenses are not verified.
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{11} \rightarrow t_{12} \rightarrow t_{13}$	After successful patient search by fingerprint, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.

continued on next page

Table 4.1: *continued*

T-**invariant**	**Remark**
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{21}$	If no patient(s) are found with search by fingerprint, name or identifier and if chosen not to register, return to homepage.
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{20} \rightarrow t_{22} \rightarrow t_{23} \rightarrow t_{24} \rightarrow t_{26} \rightarrow t_{27}$	Go back to homepage when patient registration is unsuccessful due to invalid form data.
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{31} \rightarrow t_{32} \rightarrow t_{33} \rightarrow t_{35}$	If do not confirm to add fingerprint to an existing patient, return to homepage.
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{20} \rightarrow t_{22} \rightarrow t_{23} \rightarrow t_{24} \rightarrow t_{25} \rightarrow t_{28} \rightarrow t_{12} \rightarrow t_{13}$	After successful registration of new patient into EMR, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{30} \rightarrow t_{12} \rightarrow t_{13}$	If fingerprint data already exists for patient(s) found by searching with name or identifier, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.

continued on next page

Table 4.1: *continued*

T-invariant	Remark
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{31} \rightarrow t_{32} \rightarrow t_{33} \rightarrow t_{34} \rightarrow t_{12} \rightarrow t_{13}$	If fingerprint data does not exist for the patient found by searching with name or identifier, then scan, confirm and add fingerprint, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.

4.3 Quantitative analysis

Petri nets have discrete states and are event driven. In quantitative analysis of the Petri net model, the dynamic behaviors of the model are analyzed. In other words, the different states (markings) that are reached due to occurrence of events (transitions) are analyzed. These behaviors are characterized by the movement of tokens in the Petri net due to firings of enabled transitions. Tokens are added or removed from places causing a change in the state (marking) of Petri net.

Reachability tree is a finite tree representation for infinite number of states (markings). Nodes in the tree represent states (markings) and arcs represent fired transitions. Following is the notation for construction of reachability tree.

Root node is the initial state (marking) of the Petri net.

Terminal node is the node from which no transition can fire.

Duplicate node is the node which is identical to a node that is already in the tree.

An algorithm is developed for the construction of reachability tree as described below. The algorithm identifies the reachable markings and its transition firing sequence. It also labels nodes if they are duplicate or terminal.

Algorithm for constructing reachability tree of a Petri net

Start algorithm

1. Declare and initialize the input incidence matrix B^-, incidence matrix B and initial marking M_0 of the Petri net.

2. Initialize the reachability tree T with root node M_0.

3. **While** non-duplicate and non-terminal markings (M_j) exist in reachability tree T, **do** the following for each M_j.

    4. If no transitions are enable at M_j, label M_j as 'terminal node'.

    5. **While** enabled transitions (t_i) exist at M_j, **do** the following for each t_i.

        6. Find the new marking M_k obtained by firing t_i at M_j.

        7. Add arc $M_j \xrightarrow{t_i} M_k$ to the reachability tree T.

        8. If marking M_k is identical to a marking already in the reachability tree T, label M_k as 'duplicate node'.

End algorithm

A reachability tree (Fig. 4.1) is constructed for Muzima fingerprint Petri net model using the algorithm developed.

Since this Petri net model is conservative and bounded as discussed in Section 4.2.1, and the initial marking contains a total of one token, implies all the reachable markings will also contain only one token. The markings in this reachability tree can be described as

$$M_i = [m(p_1),\ m(p_2), \cdots m(p_{27}),\ m(p_{28})]^T$$

where $m(p_i) = 1$ and all other entries are zero. From the reachability tree shown in Fig. 4.1,

- marking M_1 is the root node.

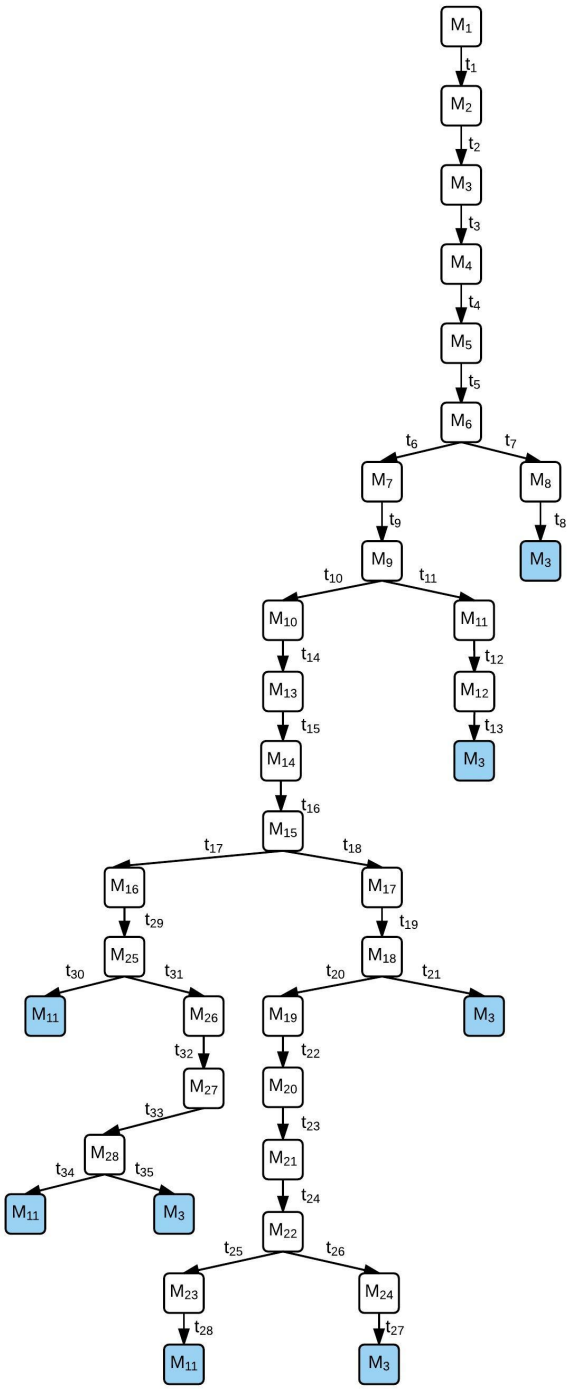

Fig. 4.1. Reachability tree for Muzima fingerprint Petri net model

- blue shaded nodes are the duplicate nodes for markings M_3 and M_{11} as labeled.
- no terminal nodes exist.
- there exists a total of 28 reachable markings.

The reachability tree analysis helps to identify if there are any overflows existing in the system (boundedness of a Petri net). All places in this Petri net model are 1-bounded since the number of tokens in every place is always less than or equal to 1 for every marking M_i that is reachable from initial marking M_0 as observed in the reachability tree. Also, this Petri net model is safe since only 0's and 1's appear in the nodes of the reachability tree.

This analysis also provides information about whether this Petri net model meets the system requirements by reaching all the desired states and not reaching any forbidden states. It also helps to determine whether this Petri net model is able to reach a particular state M_i from the initial state M_1. The sequence of transition firings which occur when the state M_1 is transformed to a state M_i, shows the functional behavior of the system. It also reflects if there are any different or unanticipated functional behavior of this system when multiple sequences of transition firings transform the state M_1 to state M_i.

4.4 Time Petri net analysis

A Time Petri net model for Muzima fingerprint module is developed in Section 3.5. Analysis can be performed on this model to determine the minimum and maximum time elapsed for the completion of T-invariant (functional loops in the system).

Consider one of the transition invariant TI of Muzima fingerprint Petri net model (Table 4.1) with the firing sequence $t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{11} \rightarrow t_{12} \rightarrow t_{13}$ which defines a loop in the system described as 'after successful patient search by fingerprint from homepage, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage'. With the help of the

minimum and maximum elapse times for these transitions defined in Table 3.8, we can analyze the minimum and maximum time taken to complete the TI as follows:

$$p_3 \xrightarrow{t_3[0,1]} p_4 \xrightarrow{t_4[0,1]} p_5 \xrightarrow{t_5[0,5]} p_6 \xrightarrow{t_6[0,3]} p_7 \xrightarrow{t_9[0,1]} p_9 \xrightarrow{t_{11}[0,2]} p_{11} \xrightarrow{t_{12}[0,1]} p_{12} \xrightarrow{t_{13}[0,1]} p_3$$

By serial fusion of transitions and their firing time ranges we get,

$$p_3 \xrightarrow{TI[0,15]} p_3 \tag{4.4}$$

From Eqn. 4.4, we can analyze the time taken to complete this transition invariant TI lies in the range [0,15]. Table 4.2 summarizes the minimum and maximum time taken to complete all the T-invariants (Table 4.1) for the Muzima fingerprint Petri net model.

Table 4.2: Summary of minimum and maximum time to complete T-invariants identified for Muzima fingerprint Petri net model.

T-invariant	**Remark**	$[t_{min}, t_{max}]$
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_7 \rightarrow t_8$	Return back to homepage when fingerprint applet licenses are not verified.	[0,10]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{11} \rightarrow t_{12} \rightarrow t_{13}$	After successful patient search by fingerprint, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.	[0,15]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{21}$	If no patient(s) are found with search by fingerprint, name or identifier and if chosen not to register, return to homepage.	[0,20]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{20} \rightarrow t_{22} \rightarrow t_{23} \rightarrow t_{24} \rightarrow t_{26} \rightarrow t_{27}$	Go back to homepage when patient registration is unsuccessful due to invalid form data.	[0,26]

continued on next page

Table 4.2: *continued*

T-**invariant**	**Remark**	$[t_{min}, t_{max}]$
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{31} \rightarrow t_{32} \rightarrow t_{33} \rightarrow t_{35}$	If do not confirm to add fingerprint to an existing patient, return to homepage.	[0,27]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{18} \rightarrow t_{19} \rightarrow t_{20} \rightarrow t_{22} \rightarrow t_{23} \rightarrow t_{24} \rightarrow t_{25} \rightarrow t_{28} \rightarrow t_{12} \rightarrow t_{13}$	After successful registration of new patient into EMR, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.	[0,28]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{30} \rightarrow t_{12} \rightarrow t_{13}$	If fingerprint data already exists for patient(s) found by searching with name or identifier, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.	[0,22]
$t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_6 \rightarrow t_9 \rightarrow t_{10} \rightarrow t_{14} \rightarrow t_{15} \rightarrow t_{16} \rightarrow t_{17} \rightarrow t_{29} \rightarrow t_{31} \rightarrow t_{32} \rightarrow t_{33} \rightarrow t_{34} \rightarrow t_{12} \rightarrow t_{13}$	If fingerprint data does not exist for the patient found by searching with name or identifier, then scan, confirm and add fingerprint, display the patient record, clicking on which the user can go to patient dashboard and then return to homepage.	[0,29]

5. SIMULATION

Simulation is important for understanding the behavior of a system. With the same idea, Muzima fingerprint Petri net model is simulated with the help of software tools named Netlab [20] and Tina [21] to analyze its properties, behavior and to gain more insight into the system. The theoretical results obtained in Chapter 4 are verified with the simulation results.

The Muzima fingerprint Petri net model with its initial marking (Fig. 3.15) is given as an input to the software named Netlab [20]. The incidence matrices for this Petri net are generated by the software and can be viewed by navigating to 'Invariants and Algebra → Net matrices and vectors'. The incidence matrix B that is generated for the Petri net model is shown in Fig. 5.1 and Fig. 5.2. This incidence matrix B generated is identical to the incidence matrix calculated theoretically as shown in Eqn. 4.3.

P-invariant(s) and T-invariant(s) are also generated by Netlab for the given input Petri net. They can be viewed by navigating to 'Invariants and Algebra → Show P-invariants and Show T-invariants'. The P-invariant(s) generated by Netlab for the Muzima fingerprint Petri net model is shown in the Fig. 5.3. This result is identical to the theoretical result obtained as shown in Section 4.2.1.

The T-invariant(s) identified by Netlab for the Muzima fingerprint Petri net model are shown in the Fig. 5.4. These eight T-invariants are equivalent those obtained theoretically as shown in the Table. 4.1.

The reachability tree construction of a Petri net as dicussed in Section 2.4.5 could become exhaustive with increase in size of Petri net (increase in number of places and transitions). To simplify the construction of reachability tree, a MATLAB program is developed (as described below) which identifies the reachable markings, their transition firing sequence and tags them if they are duplicate or terminal. The inputs given

to this MATLAB program to obtain the reachability tree for Muzima fingerprint Petri net model are its input incidence matrix B^-, incidence matrix B and initial marking M_0.

MATLAB program

```
%MATLAB program to list all the reachable states of a given Petri net model and tag any duplicate or terminal nodes if exist
%Author: Archana Eadara

%Initial Marking is added as the first column of reachable markings matrix with index 1
M(:,1) = M0;
%n denotes index of next reachable marking
n = 2;
%j denotes the index of marking in consideration from which enabled transitions are fired to obtain new reachable states
j = 1;

%loop to find all reachable states from initial marking until only terminal or duplicates nodes are left in the reachability tree
while j <= size(M,2)
t = 1;
Temp = zeros(28, 1);

    %check which of 35 transitions can be enabled and then fire them to obtain a new marking
    while t <= 35
        if (M(:,j) >= BMinus(:,t))
            Temp = M(:,j) + B(:,t);
            isduplicate = 0;

            %Check if the new marking is duplicate
```

```
23              for i = 1:n-1
24                  if(M(:,i) == Temp)
25                      isduplicate = 1;
26                      duplicateIndex = i;
27                  end
28              end
29
30              %if it is not duplicate, add the new marking to the set
                    ↪ of reachable markings
31              if(isduplicate == 0)
32                  M(:,n) = Temp;
33                  fprintf('M[%i]--t%i->M[%i] \n',j, t, n)
34                  n = n + 1;
35              else
36                  fprintf('M[%i]--t%i->M[%i] (duplicate node) \n',j, t,
                        ↪  duplicateIndex)
37              end
38          end
39          t = t + 1;
40      end
41
42      %check if the new marking is terminal node
43      if(Temp == zeros(28, 1))
44          fprintf('M[%i] is terminal node\n', j)
45      end
46      j = j + 1;
47  end
48
49  %M denotes matrix with reachable markings added as it columns
50  fprintf('\nThe number of reachable markings are %i which are added as
        ↪  columns to matrix M\n', size(M,2))
51  M
```

The reachability tree for the Muzima fingerprint Petri net model generated from executing the MATLAB program can be described as follows where each line in the

form $M[i] - -tj- > M[k]$ represents an arc (transition tj fired) of the reachability tree drawn from the marking stored in column i of matrix M to the marking stored in column k of matrix M.

```
M[1]--t1->M[2]
M[2]--t2->M[3]
M[3]--t3->M[4]
M[4]--t4->M[5]
M[5]--t5->M[6]
M[6]--t6->M[7]
M[6]--t7->M[8]
M[7]--t9->M[9]
M[8]--t8->M[3] (duplicate node)
M[9]--t10->M[10]
M[9]--t11->M[11]
M[10]--t14->M[12]
M[11]--t12->M[13]
M[12]--t15->M[14]
M[13]--t13->M[3] (duplicate node)
M[14]--t16->M[15]
M[15]--t17->M[16]
M[15]--t18->M[17]
M[16]--t29->M[18]
M[17]--t19->M[19]
M[18]--t30->M[11] (duplicate node)
M[18]--t31->M[20]
M[19]--t20->M[21]
M[19]--t21->M[3] (duplicate node)
M[20]--t32->M[22]
M[21]--t22->M[23]
M[22]--t33->M[24]
M[23]--t23->M[25]
M[24]--t34->M[11] (duplicate node)
M[24]--t35->M[3] (duplicate node)
M[25]--t24->M[26]
```

```
M[26]--t25->M[27]

M[26]--t26->M[28]

M[27]--t28->M[11] (duplicate node)

M[28]--t27->M[3] (duplicate node)

The number of reachable markings are 28 which are added as columns to matrix M

M =

1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
```

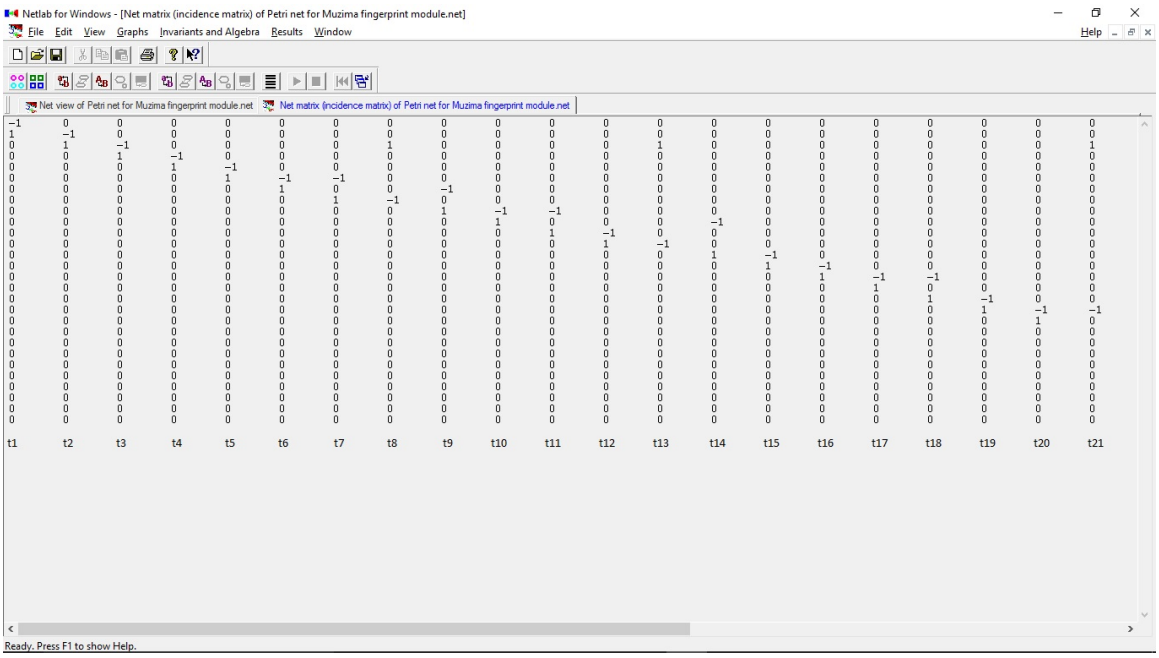

Fig. 5.1. Incidence matrix B generated by Netlab for the Muzima fingerprint Petri net model

The reachability tree generated from this MATLAB program is identical to the reachability tree generated theoretically as shown in Fig. 4.1.

To simulate the Muzima fingerprint Time Petri net model (Fig. 3.16), a software tool named Tina [21] is used. The Time Petri net model is given as an input to the software. To check the time taken by a T-invariant (Table. 4.1) or the time taken to reach a state, launch the stepper simulator by navigating to 'Tools $\rightarrow$ stepper simulator'. Use the 'min', 'max', '-1' and '$+1$' buttons to apply the desired delay (within the range defined by α and β)for an enabled transition to fire. The total time taken to complete a T-invariant or to reach a specific reachable state is displayed on the right as highlighted in the Fig. 5.5. For instance, the time taken to complete the T-invariant described with firing sequence $t_3 \rightarrow t_4 \rightarrow t_5 \rightarrow t_7 \rightarrow t_8$ with the maximum possible delays is obtained as 10.0 (Fig. 5.5) which is identical to the theoretical results obtained as shown in the Table. 4.2

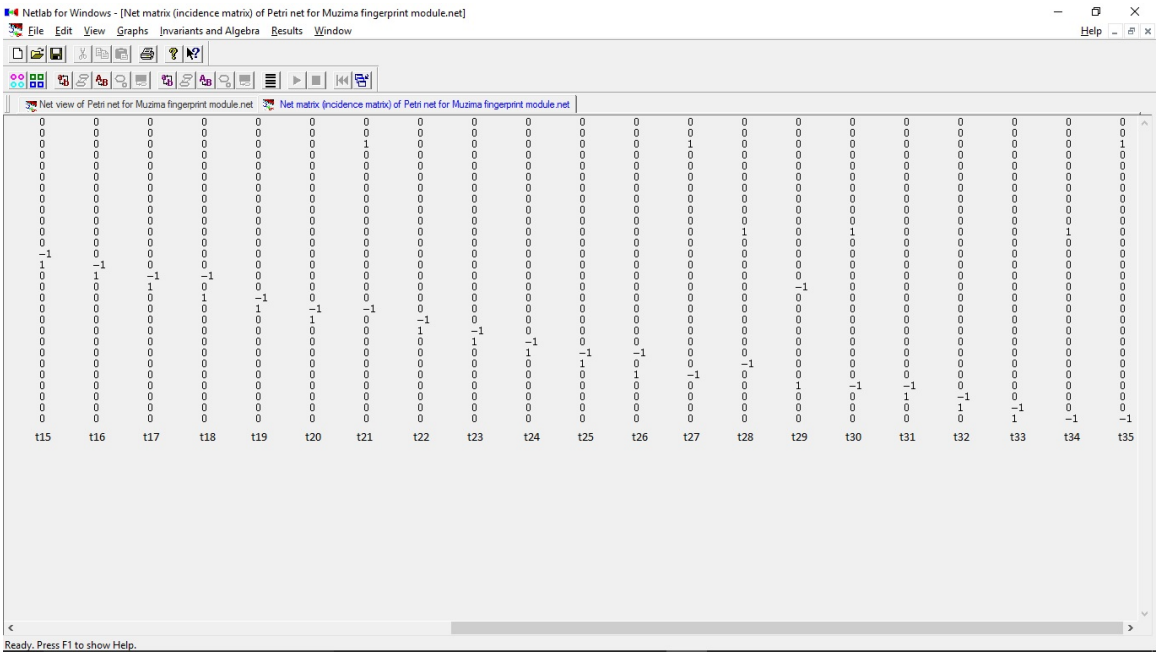

Fig. 5.2. Incidence matrix B generated by Netlab for the Muzima fingerprint Petri net model (Contd.)

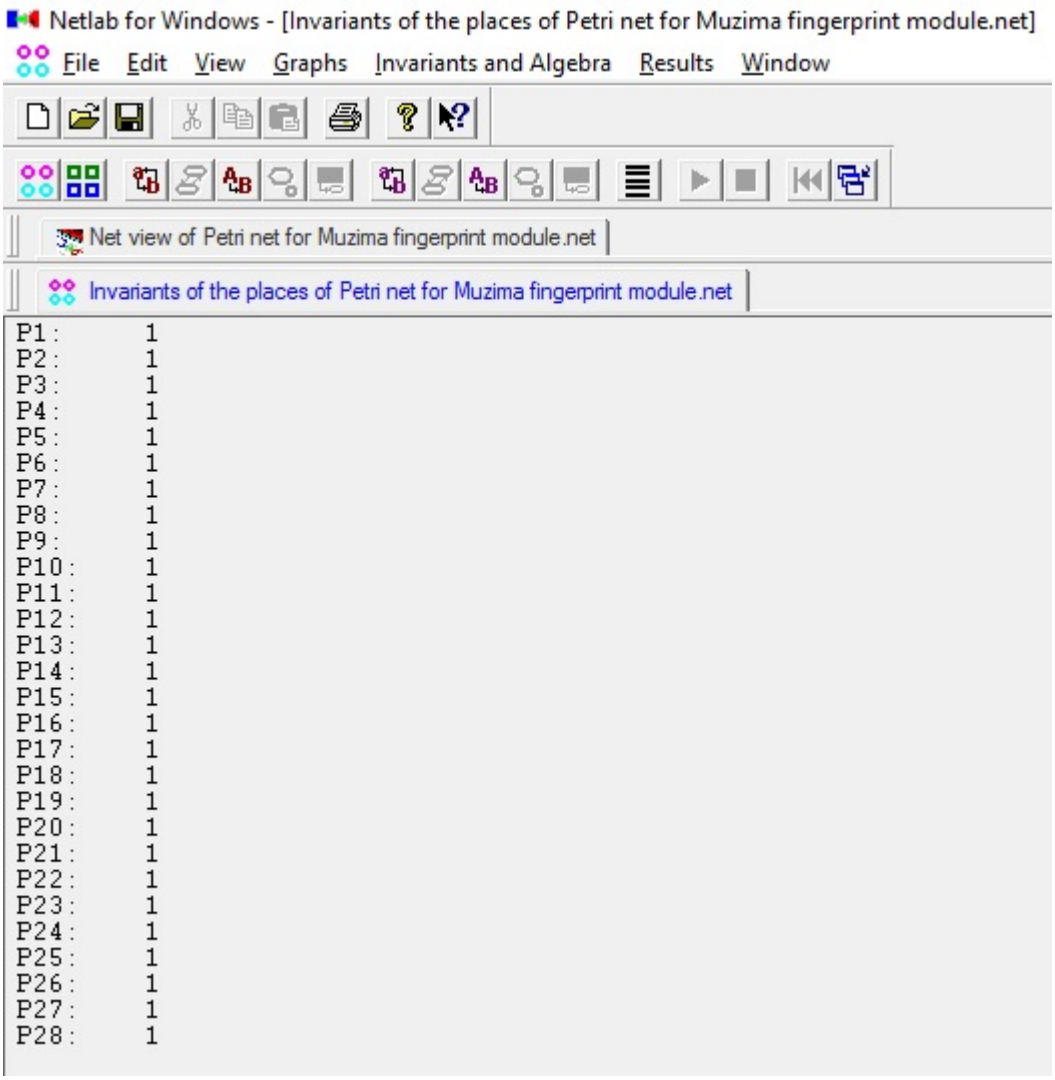

Fig. 5.3. *P*-invariant generated by Netlab for the Muzima fingerprint Petri net model

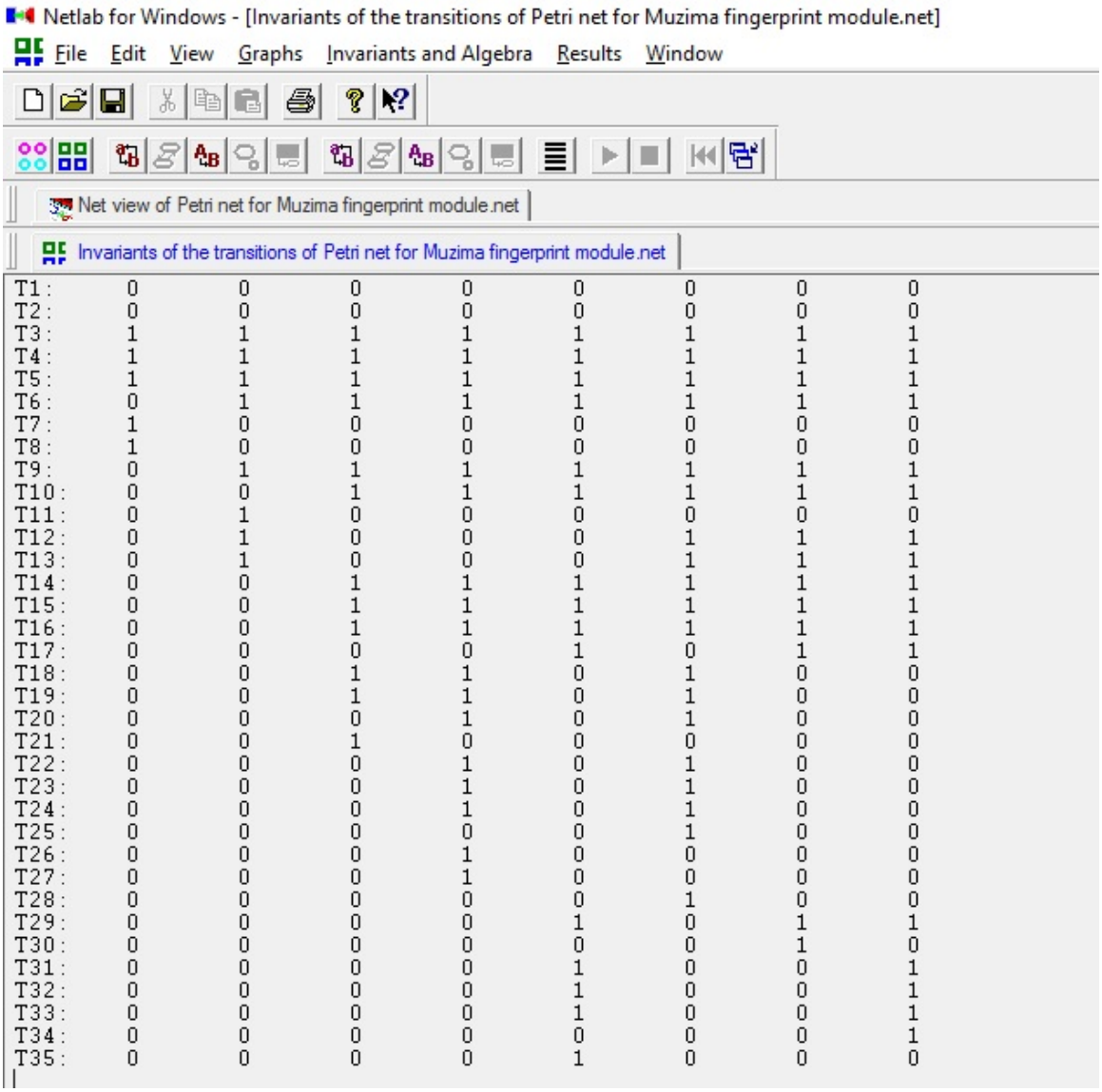

Fig. 5.4. T-invariant generated by Netlab for the Muzima fingerprint Petri net model

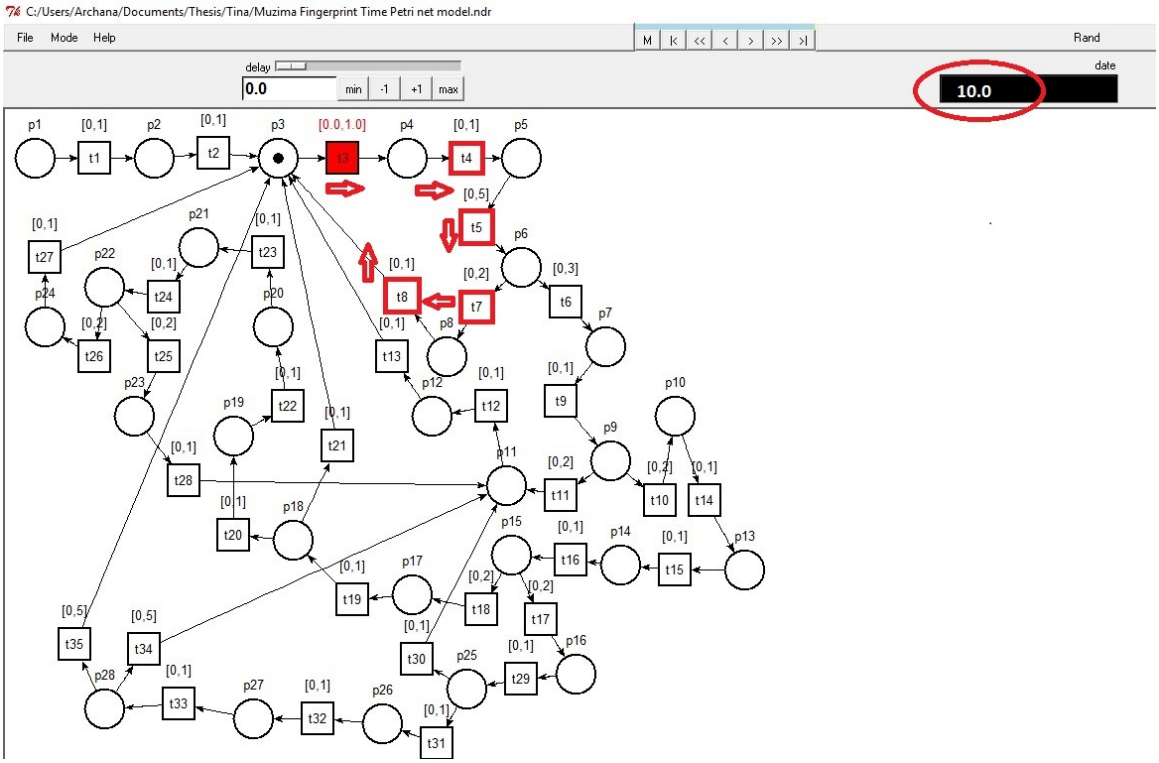

Fig. 5.5. Simulation of Muzima fingerprint Time Petri net model

6. CONCLUSION

The thesis herein presented both Ordinary and Time Petri net models for the workflow of Muzima fingerprint module focusing on the functional behavior of the system. These developed Petri net models are able to identify the qualitative (structural behavior) and quantitative (dynamic behavior) characteristics of the system. It analyzed the place invariants (token conservation) and transition invariants (functional loops) present in the system modeled. The Time Petri net model developed identifies the time taken to reach a particular state or to accomplish a fuctional behavior in the system. Also the reachability tree analysis provided insight about all the possible reachable states in the system. The generic MATLAB program developed for the reachability tree analysis also supports the theoretical results.

6.1 Future work

Since Petri nets complement analysis of healthcare workflow, further application of Petri net modeling will contribute to the analysis, development and testing of the healthcare software. By assigning weights to the arcs in the Petri net model, will help us better analyze the weight (resources) inflow and outflow requirements. A more deeper workflow analysis can be performed for this fingerprint module which can provide us better insight about the system for its performance improvement. Another potential work can be performed on reducing the state space size of the Time Petri net model by serial and lateral fusion of the Time Petri net model to identify time delays in the system.

LIST OF REFERENCES

LIST OF REFERENCES

[1] J. Lynn, "The challenge of patient identification and patient matching," March 11, 2016. Internet Source. Available at: http://www.emrandhipaa.com/emr-and-hipaa/2016/03/11/the-challenge-of-patient-identification-and-patient-matching/ (Last Date Accessed: April 20, 2016).

[2] C. of Healthcare Information Management Executives, "Summary of chime survey on patient data-matching," May 16, 2012. Internet Source. Available at: https://chimecentral.org/wp-content/uploads/2014/11/Summary_of_CHIME_Survey_on_Patient_Data.pdf (Last Date Accessed: April 20, 2016).

[3] E. Joffe, C. F. Bearden, M. J. Byrne, and E. V. Bernstam, "Duplicate patient records–implication for missed laboratory results," in *AMIA Annual Symposium Proceedings*, vol. 2012, p. 1269, American Medical Informatics Association, 2012.

[4] OpenMRS, "Openmrs releases 2015 annual report," February 25, 2016. Internet Source. Available at: http://openmrs.org/2016/02/openmrs-releases-2015-annual-report/ (Last Date Accessed: April 20, 2016).

[5] C. Mahulea, L. Mahulea, J.-M. Garcia-Soriano, and J.-M. Colom, "Petri nets with resources for modeling primary healthcare systems," in *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference*, pp. 639–644, IEEE, 2014.

[6] W. Stis, C. Maiga, L. Hwei-Nung, and H. Jia-Sheng, *Proceedings of the 7th WSEAS International Conference on E-ACTIVITIES*, ch. Design Petri Net to Simulate the Processes of Tele-Healthcare, pp. 183–188. WSEAS Press, 2008.

[7] S. S. Choi, M. K. Choi, W. J. Song, and S. H. Son, "Ubiquitous rfid healthcare systems analysis on physionet grid portal services using petri nets," in *2005 Fifth International Conference on Information, Communications and Signal Processing*, pp. 1254–1258, IEEE, 2005.

[8] J. Nam, "A trust framework of ubiquitous healthcare with advanced petri net model," in *Electronic Healthcare*, pp. 122–129, Springer, 2008.

[9] M. Jansen-Vullers and H. Reijers, "Business process redesign at a mental healthcare institute: A coloured petri net approach," in *Proceedings of the Sixth Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools (PB-576)*, pp. 21–38, 2005.

[10] S. Mtibaa and M. Tagina, "Managing changes in citizen-centric healthcare service platform using high level petri net," *arXiv preprint arXiv:1210.5516*, 2012.

[11] S. Mtibaa and M. Tagina, "Timing constraints support on petri-net model for healthcare system design," *arXiv preprint arXiv:1210.5374*, 2012.

[12] C. A. Petri and W. Reisig, "Petri net," *Scholarpedia*, vol. 3, no. 4, p. 6477, 2008.

[13] C. G. Cassandras and S. Lafortune, *Introduction to discrete event systems*. Springer Science & Business Media, 2009.

[14] P. M. Merlin and D. J. Farber, "Recoverability of communication protocols–implications of a theoretical study," *IEEE Transactions on Communications*, vol. 24, no. 9, pp. 1036–1043, 1976.

[15] G. Bucci and E. Vicario, "Compositional validation of time-critical systems using communicating time petri nets," *IEEE Transactions on Software Engineering*, vol. 21, no. 12, pp. 969–992, 1995.

[16] Y. Deng, J. Wang, and R. Sinha, "Integrated architectural modeling of real-time concurrent systems with applications in fms," in *in FMS, Proceedings of the 10th International Conference on Software Engineering and Knowledge Engineering*, Citeseer, 1997.

[17] Y. Deng, J. Wang, and R. Sinha, "Incremental architectural modeling and verification of real-time concurrent systems," in *Second International Conference on Formal Engineering Methods, 1998. Proceedings.*, pp. 26–34, IEEE, 1998.

[18] J. J. Tsai, S. J. Yang, and Y.-H. Chang, "Timing constraint petri nets and their application to schedulability analysis of real-time system specifications," *IEEE Transactions on Software Engineering*, vol. 21, no. 1, pp. 32–49, 1995.

[19] J. Wang, *Timed Petri nets: Theory and application*, vol. 9. Springer Science & Business Media, 2012.

[20] "Netlab," *IRT*, January, 2008. Internet Source. Available at: http://www.irt.rwth-aachen.de/en/for-students/downloads/petri-net-tool-netlab/ (Last Date Accessed: April 20, 2016).

[21] "Tina," *LAAS*, November, 2004. Internet Source. Available at: http://projects.laas.fr/tina//home.php (Lat Date Accessed: April 20, 2016).