Formal Verification of the Health Code System for Quarantine Management: A Case Study in China

By

Shahid Ali Khan

(Registration No: 329695)

Department of Computer Science

School of Electrical Engineering and Computer Science (SEECS)

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

(2024)

Formal Verification of the Health Code System for Quarantine Management: A Case Study in China

By

Shahid Ali Khan

(Registration No: 329695)

A thesis submitted to the National University of Sciences and Technology, Islamabad,

in partial fulfillment of the requirements for the degree of

Master of Science in
Computer Science

Supervisor: Dr. Aimal Tariq Rextin

School of Electrical Engineering and Computer Science (SEECS)

National University of Sciences & Technology (NUST)

Islamabad, Pakistan

(2024)

THESIS ACCEPTANCE CERTIFICATE

Certified that final copy of MS Thesis entitled "Formal Verification of the Health Code System for Quarantine Management: A Case Study in China" written by Mr. Shahid Ali Khan (Registration No. 329695), of SEECS has been vetted by undersigned, found complete in all respects as per NUST Statutes/ Regulations/ Masters Policy, is free of plagiarism, errors, and mistakes and is accepted as partial fulfillment for award of MS/M Phil degree. It is further certified that necessary amendments as pointed out by GEC members and evaluators of the scholar have also been incorporated in the said thesis.

Signature: [illegible]

Name of Supervisor Dr. Aimal Tariq Rextin

Date: 01-Jul-2024

Signature (HOD): [illegible]

Date: 01-Jul-2024

Signature (Dean/ Principal) [illegible]

Date: 01-Jul-2024

Form TH-4

<u>National University of Sciences and Technology</u>

MS THESIS WORK

We hereby recommend that the thesis prepared under our supervision by: Mr. Shahid Ali Khan (329695) Titled: Formal Verification of the Health Code System for Quarantine Management: A Case Study in China be accepted in partial fulfillment the requirements for the award of MS degree.

<u>Examination Committee Members</u>

1.Name Dr. Sohail Iqbal Signature: [illegible]

2.Name Ms. Iram Tariq Bhatti Signature: [illegible]

Supervisor's Name Dr. Aimal Tariq Rextin Signature: [illegible]

[illegible]

Dr. Muhammad Imran Malik

HoD / Associate Dean

9-August-2024

Date

<u>COUNTERSIGNED</u>

09-August-2024

Date

[illegible]

Muhammad Ajmal Khan

Principal

AUTHOR'S DECLARATION

I Shahid Ali Khan hereby state that my MS thesis titled "Formal Verification of the Health Code System for Quarantine Management: A Case Study in China" is my own work and has not been submitted previously by me for taking any degree from National University of Sciences and Technology, Islamabad or anywhere else in the country/ world.

At any time if my statement is found to be incorrect even after I graduate, the university has the right to withdraw my MS degree.

Name of Student: Shahid Ali Khan

Date: ______________________________

DEDICATION

To my Parents, teachers, wife and loyal friends.

ACKNOWLEDGEMENTS

Above all, the heartfelt appreciation is as follows: First of all, the immense gratitude is expressed to the supervisor Dr. Aimal Tariq Rextin, Ex-supervisor Dr. Adnan Rashid and Ms. Iram Tariq Bhatti for the constant encouragement, essential supervision and inspiration shaping the given project. I would also like to thank Mr. Osama Raza for all the help and assistance you have provided myself throughout all these years it has been incredibly useful. Last but not least, I would like to extend my heartfelt gratitude to my parents and my wife for their great understanding and patience in the entire process.

TABLE OF CONTENTS

LIST OF TABLES

Page No.

LIST OF FIGURES

LIST OF SYMBOLS, ABBREVIATIONS AND ACRONYMS

HCA’s	Health Code Apps
STA	Stochastic Timed Automata [1]
QMS	Quarantine Management System
HA	Hybrid Automata
PTA	Probabilistic Timed Automata [2]
MA	Markov Automata [[3]
TA	Timed Automata
MDP	Markov Decision Processes
CTMDP	continuous Time MDP
LTS	Labelled Transition Systems
DTMC	discrete-Time Markov Chains
CTMC	Continuous-Time Markov Chains

ABSTRACT

The Novel Coronavirus (COVID-19), which emerged in late 2019, was first identified in Wuhan, China. It rapidly escalated into a global pandemic, affecting millions and causing unprecedented social and economic disruptions. In response, countries worldwide adopted various containment strategies, including lockdowns, social distancing, and mandatory masks. Notably, China's rigorous Quarantine Management System (QMS) was instrumental in curbing the virus's spread. This system enforced mandatory quarantines, utilized contact tracing, and established centralized quarantine facilities for confirmed cases. Technological advancements, such as health QR codes and facial recognition, were pivotal in monitoring individuals and ensuring adherence to quarantine protocols. The system's effectiveness in containing the virus was notable and was widely mentioned in media worldwide.

To ensure the reliability and effectiveness of the health code app, formal verification methods such as model checking were employed. The PRISM model checker, a state-of-the-art probabilistic model checker was utilized to perform formal verification of the app's algorithms. This process involved creating a detailed mathematical model of the app's behavior and using PRISM to verify that the app adhered to the desired properties of accuracy and efficiency. The use of PRISM allowed for a rigorous analysis of the health code app, ensuring that it functioned correctly within the QMS.

Keywords: COIVD-19; Pandemic, PRISM model checker, Chinese Quarantine Management System (QMS)

CHAPTER 1: INTRODUCTION

In December 2019, a novel coronavirus, later named COVID-19, emerged from the city of Wuhan in China's Hubei province, marking the beginning of an unprecedented global health crisis. The virus, characterized by its highly contagious nature, rapidly transcended borders, leading to the World Health Organization declaring it a pandemic. This thesis looks at the many ways the COVID-19 pandemic affected the world, how different countries responded, and the new technologies created to combat this crisis.

The pandemic's reach was indiscriminate, affecting individuals and communities across every continent. As the virus spread, it laid bare the vulnerabilities of even the most robust healthcare systems, disrupted economies, and altered the very fabric of society. Governments worldwide were compelled to implement stringent measures, ranging from complete lockdowns to social distancing and mandatory masks, in an effort to control the viral transmission.

A focal point of this thesis is the analysis of China's Quarantine Management System (QMS)[1], which became a cornerstone in the country's response to the pandemic. The QMS leveraged technology, including a health code app, to monitor and regulate the movement of people, thereby effectively managing the spread of the virus.

The Chinese health code system, as shown in Figure 1.1, also known as the Green Health QR Code, is a digital platform that was launched in early 2020 to track and control the spread of COVID-19 in China. The main features of the app includes health declaration in which user is required to fill out a health declaration form which includes their travel history, current health status, and other relevant information. The data is collected and analyzed by the system to determine the user's risk of being infected with COVID-19. The system generates a QR code based on the user's health status, which is color-coded based on the user's risk level. Green means low risk, yellow means medium risk, and red means high risk. The QR code is used to verify a person's health status when entering public places

[1] Health QR code system implemented in Shanxi (chinadaily.com.cn)

such as public transportation, shopping malls, and restaurants. The system allows for the sharing of health information between different departments and agencies, such as health authorities, transportation authorities, and law enforcement agencies. This enables the government to track the movement of individuals who may have been exposed to the virus and to quickly respond to any outbreaks. The system is enforced by law, and individuals who are found to have provided false information or who have violated quarantine rules may face fines or other penalties.

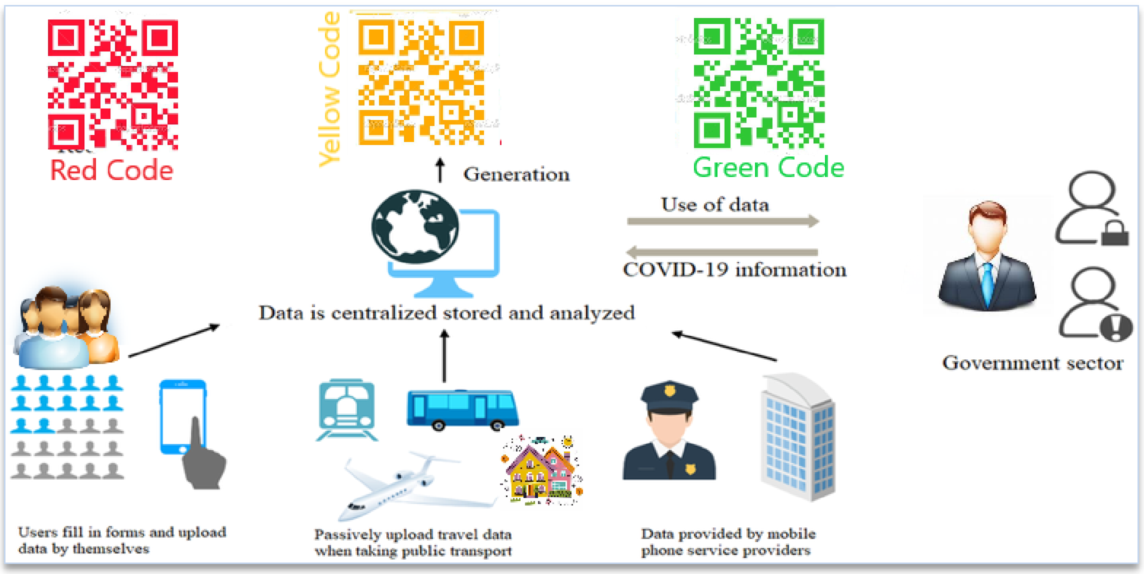

Red Code
Yellow Code
Green Code
Generation
Use of data
COVID-19 information
Data is centralized stored and analyzed
Government sector
Users fill in forms and upload data by themselves
Passively upload travel data when taking public transport
Data provided by mobile phone service providers

Figure 1.1 Color Guideline of Shanxi Health Code[2], Green Health Code App an example

1.1 Motivation

COVID-19 Pandemic once again demonstrated how crucial is the involvement of technology while combating such challenges to public health. The most important technological innovations that were employed was the use of the Green health code app from China where the virus was first discovered, to track and contain the virus to avoid further spread. However, it is far from being insignificant since its efficiency and actual applicability depends upon the realization of correctly functioning algorithms that occupy a significant part in the contemporary technological world.

[2] China’s Novel Health Tracker: Green on Public Health, Red on Data Surveillance | Trustee China Hand | CSIS

Given the high-stakes nature of these applications, which can significantly impact public life and economy, it is imperative to ensure their flawless operation. Governments and public health agencies cannot afford the risk of system failures in tools designed to protect citizens. This thesis is driven by the necessity to develop rigorous verification methodologies to instill confidence in the algorithms powering these critical systems. By ensuring the correctness and reliability of these algorithms, we can maximize the benefits of digital health solutions while minimizing potential risks.

Moreover, the complexity and dynamism of real-world environments, coupled with the potential for adversarial attacks, necessitate robust verification techniques. Traditional testing methods often fall short in guaranteeing system correctness, especially in the context of safety-critical applications. Formal verification offers a systematic and mathematically grounded approach to identify potential vulnerabilities and ensure system reliability. This research aims to contribute to the development of advanced verification techniques tailored to the unique challenges posed by public health applications.

The motivation of this thesis is the critical nature of the system, that there should be no chance the health code app fails to work as designed under certain circumstances.

CHAPTER 2: LITERATURE REVIEW

The overall literature has been shortlisted after thoroughly searching relevant papers of renowned publishers. The shortlisted papers are mostly related to health-code apps and Model checking with PRISM.

2.1 Literature Review

Here in this section, we will discuss the related work done so far. The first part covers details of Chinese Health code App and its role in the society, the second part contains the verification techniques used for health-related applications and third part is about the use of formal verification specially, model checking for different systems.

The Health Code app has been effective in controlling the spread of COVID-19 by identifying and managing individuals' exposure risks through color-coded statuses [4], [5], [6]. The app has allowed China to restore social and economic activities by providing a reliable method for monitoring and controlling the movement of people [6]. Public opinion on the Health Code app is mixed, with a majority supporting moderate use post-pandemic, while a significant minority supports its expansive use for broader purposes [5]. Trust in government and perceived convenience are major factors influencing public acceptance of the app [7]. The Health Code app is seen as a tool for digital surveillance, extending beyond pandemic control to broader governance and population control measures [8], [9]. The app's integration into daily life has led to its gradual acceptance as part of the digital infrastructure, despite ongoing concerns about privacy and surveillance [9]. There is a push from both government and tech firms to expand the use of the Health Code app beyond pandemic control, potentially normalizing its use for various public health and governance purposes [5], [10]. The app's role in future health surveillance and governance will depend on addressing and ensuring informed consent from users[11].

The paper by Fan Liang examines the Health Code initiative as a case study of digital platforms[4] involvement in health surveillance and the management of the COVID-19 pandemic in China. The paper discusses the rapid implementation of contact-tracing apps worldwide, with a focus on China's Health Code system developed by Alipay and WeChat, it highlights the Health Code's role in assessing contagion risks based on travel history, time spent in risky areas, and contact with potential carriers. It suggests that the Health Code represents a shift towards platform-mediated visibility of citizens during a health crisis. The paper posits that the adoption of tracing apps like Health Code may become a standard practice for health surveillance in many countries. This indicates a potential future where digital platforms could play a significant role in global health governance.

The paper introduces HCAs (Health code App’s) as more than just tracking tools [4]; it is a part of China's digital governance and reshape state-society relations through digital and human surveillance. Most people support moderate use of Health Code Apps after the pandemic, with many also supporting broader use, showing cautious openness to technology. The paper concludes that HCAs may evolve into an integral part of China's digital infrastructure with significant social implications.

Model checking is a formal verification technique that systematically checks whether a model of a system meets a given specification. In the realm of health code apps, model checking is employed to validate the correctness of the app's algorithms and their implementation. Büyükkaramikli et al. [12] introduce the TECH-VER (Technical verification) checklist, which aims to reduce errors in models and improve their credibility. This checklist provides a structured approach to verifying the technical implementation of health economic decision analytical models, ensuring their correctness and effectiveness model checking is a formal method used to verify the correctness of systems with respect to certain specifications. In the context of health code apps, model checking ensures that the app behaves as expected under various scenarios, including user interactions and data processing. The PRISM model checker, a probabilistic model checker, is particularly suited for this task due to its ability to handle systems exhibiting random behavior, which is often the case with health-related applications.

A systematic approach to evaluating health apps is essential to ensure their quality and effectiveness. Gasteiger et al. [13] provide a methodological guide for conducting commercial smartphone health app reviews, introducing the TECH approach to developing review questions and eligibility criteria. Similarly, a scoping review by Lagan, Sandler, and Torous (2023) [14] identifies 45 frameworks for assessing health apps, highlighting the variability in target users and conditions.

Wiep van et al.[15] proposes an intra-host SARS-CoV-2 (severe acute respiratory syndrome coronavirus 2) dynamics model to assess the effectiveness of testing and quarantine strategies for incoming travelers, contact person management, and de-isolation. The authors highlight the importance of a combination of testing, quarantine, and isolation measures to reduce the spread of SARS-CoV-2. Alberto et al. [16] presents a mathematical model to investigate the impact of testing, contact tracing, and household quarantine on second waves of COVID-19. The authors suggest that a combination of targeted testing, contact tracing, and household quarantine could effectively reduce the transmission of SARS-CoV-2. Marcel et al. [17]discusses the impact of testing, contact tracing, and isolation on the COVID-19 epidemic in Switzerland. The authors highlight the importance of early testing, rapid contact tracing, and effective isolation measures in controlling the spread of SARS-CoV-2. Fan Yang et al.[18] emphasizes the complexities of the space and goals of these systems, and their implications for larger systems of democratic and economic control.

The effectiveness of model checking in medical applications extends beyond theoretical validation; it has practical implications for patient care and medical decision-making. Martins et al. [19] propose a framework for online verification of medical critical intelligent systems through model checking. Their work demonstrates the application of model checking to qualify medical risk scores, thereby aiding medical teams in making better-informed decisions.

The field of software model checking has evolved significantly over the past two decades. An overview of the development of software model checking highlights the integration of static analysis, model checking, and deduction to verify properties of critical systems. This evolution underscores the importance of model checking in ensuring the correctness and effectiveness of software systems, including health code apps effectiveness of model checking in medical applications extends beyond theoretical validation; it has practical implications for patient care and medical decision-making.

Usman Pervez et al. [20] employed two formalisms; the Markov Decision Process (MDP) and Continuous Time Markov Chain (CTMC) to determine failure probabilities of the e-health system under the FHIR (Fast Healthcare Interoperability Resources) standard, utilizing the PRISM model checker. The system was employed in hospitals and from the analysis process it assisted in the determination of the reliability of the system. The study of J. Smith et al.[21] is based on the PRISM conceptual framework and corresponding PRISM tools intended for the development, improvement, and assessment of the RHIS (Routine Health Information System) practice. It focuses on improving the performance of the RHIS through enhanced data quality, and its utilization.

The literature underscores the importance of model checking in the development and validation of health code apps. The PRISM model checker, in particular, stands out as a robust tool for ensuring the correctness and effectiveness of these apps. As health code apps continue to play a vital role in public health management, the insights from model checking research will be invaluable in guiding their development and ensuring their reliability. software model checking has evolved significantly over the past two decades. An overview of the development of software model checking highlights the integration of static analysis, model checking, and deduction to verify properties of critical systems. This evolution underscores the importance of model checking in ensuring the correctness and effectiveness of software systems, including health code apps effectiveness of model checking in medical applications extends beyond theoretical validation; it has practical implications for patient care and medical decision-making.

Kwiatkowska M et al. [22]outline the significant contingency in the subject of probabilistic model checking with particular emphasis on the model checking and strategic reasoning processes in PRISM. It examines its relevance, and this part of the study provides examples of autonomous systems that could relate to the framework. In one another paper[23], they focused on its features regarding continuous-time Markov chains and Markov reward models.

The QMS model by Ahmed et al. (a thesis) [27] is a probabilistic model designed to simulate a quarantine management system within a single hospital. It focuses on tracking patient health conditions and bed availability for a specific set of diseases. While the model uses statistical methods to estimate the likelihood of various events, its scope is limited to a single hospital and a predetermined set of diseases and users.

In conclusion, the Chinese health code system, while indispensable in contemporary life, necessitates stringent verification to maintain public trust and safety. This study has explored the potential of model checking, specifically using PRISM, as a rigorous approach to analyzing the system's behavior. By applying model checking techniques, we aim to identify potential vulnerabilities, enhance system reliability, and contribute to the development of more secure and trustworthy health-related applications.

2.2 Problem Statement

"The Chinese Quarantine Management System (QMS) also known as Green Health-code system has been effective in controlling virus spread and it has a safety critical nature. Despite of this fact that there are potential loopholes[3]. The challenge is to evaluate and analyze the QMS's safety, correctness, effectiveness, reliability and reachability in managing infectious diseases like COVID-19."

[3] COVID-19 and healthcare system in China: challenges and progression for .... https://globalizationandhealth.biomedcentral.com/articles/10.1186/s12992-021-00665-9.

CHAPTER 3: BACKGROUND MATERIAL

This chapter aims at developing an understanding and appreciation of the Probabilistic model checking as well as the PRISM model checker. It provides also a general description of the algorithm of the Chinese health code system that is formally verified in the thesis.

3.1 Probabilistic Model Checking

Probabilistic Model checking[24] is one of the techniques of formal verification process which aim at the use of tools in checking the correctness of a system with pre-defined characteristics. Since then it has been used widely in theory and modeling of architectures of embedded hardware-software and communication systems. The existing model checking procedures are designed and used usually for deterministic systems, where the system behavior is fully determined.

Systems exhibiting the probabilistic behavior can be modeled using various formalisms like Discrete Time Markov Chains (DTMC's), Continuous Time Markov Chains (CTMC's) Markov Decision Process (MDP) and Probabilistic Timed Automata (PTA's). While DTMC describes a direct transition between states with the probability label attached to the arcs, CTMC informs not only the probability of making the transitions from state to state but also the time delays occurred while making the transitions. These random delays are expressed through exponential probability distributions. While MDPs are DTMC with non-deterministic transitions, PTAS are CTMC with non-deterministic transitions[25]. illustrates the distinction between the aforementioned Markov models.

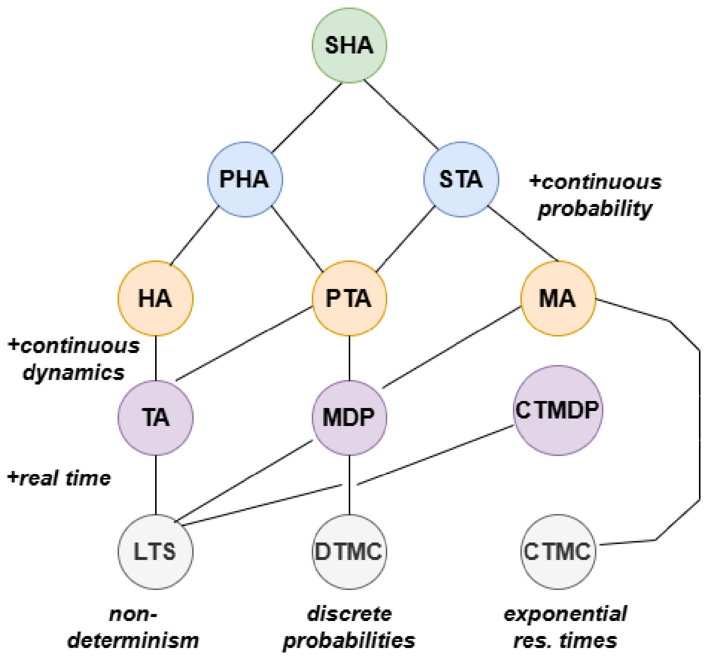
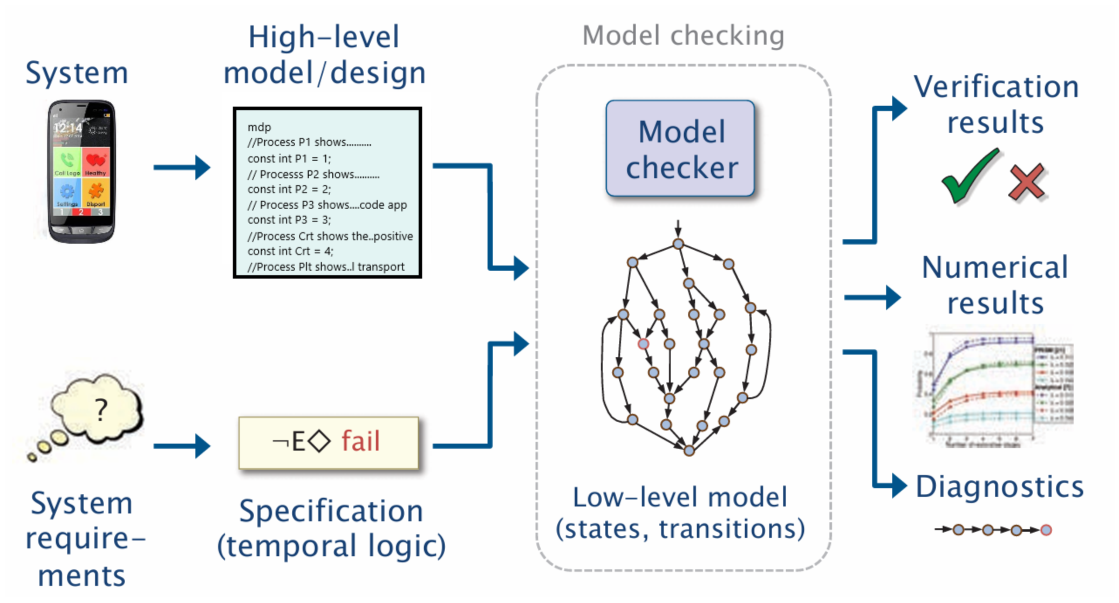


Figure 3.1: The family tree of automata-based quantitative formalisms

When the markovian model of the system under verification is exercised and standardized, the probabilistic characteristics of that system are determined formally. Probabilistic model checking is mainly based on the specification language of Probabilistic Linear Temporal Logic (PLTL). The Markovian model and probabilistic property of the system is the next to be translated into the language of the probabilistic model checker and submitted to the model checking tool. The tool carries out an analysis of the model up to its degrees of freedom ensuring that all the possible executions have been covered then the queries are solved through numerical solutions methods [24], [26].

There are a host of probabilistic model checking tools, and each of them is outstanding for one or several application areas [26]. For instance INFAMY[27] is specifically designed for model checking of innite-state CTMCs whereas PARAM [28] is used for the parametric probabilistic model checking of DTMCs. The first three model checkers, namely PASS [29] and RAPTURE [30], are intended to work with the Markov decision processes only. The Fortuna [31] model checker evaluates maximum probabilistic reachability for PTAs, and reward bounds properties of linearly priced PTAs. PRISM on the other hand supports model checking for every given in Figure 3.1 markovian model. It is a generic tool and hence we think it is appropriate for our research objectives.



```
mdp
//Process P1 shows.........
const int P1 = 1;
// Processs P2 shows.........
const int P2 = 2;
// Process P3 shows....code app
const int P3 = 3;
//Process Crt shows the..positive
const int Crt = 4;
//Process Plt shows..l transport
```

Figure 3.2: An overview of the model-checking framework with PRISM Model Checker

3.2 Markov Decision Processes

Markov Decision Processes (MDPs) are often viewed as an extension of Discrete Time Markov Chains (DTMCs). MDPs have the ability to encapsulate both nondeterministic and probabilistic behaviors. Nondeterminism is a powerful tool for modeling concurrent systems, and MDPs enable us to represent the actions of multiple probabilistic systems functioning simultaneously [32]. Nondeterminism proves to be beneficial when the precise probability of a transition is either unknown or deemed insignificant. An MDP is defined as a tuple ((S,$\bar{s}$,Steps,L), where:

- S is a finite set of states
- $\bar{s}$ ∈ S is the initial state
- Steps : S →2^Distinct(S)   is the transition function
- L: S →2^APis the labeling function

3.3 The PRISM Model Checker and its Language

In this section, we delve into the practical specification of probabilistic models using PRISM, probabilistic model checker[32]. Real-life applications often yield models with an extensive number of states, making it unfeasible to explicitly list each state and transition. To address this, we employ a high-level specification formalism that captures the essence of these models. Examples of such probabilistic models include stochastic process algebras and stochastic Petri nets. PRISM’s property specification language encompasses well-known probabilistic temporal logics, including PCTL, CSL, probabilistic LTL, and PCTL. This enables us to analyze various quantitative properties of the models, such as probabilities of failure, expected queue sizes, and termination times.

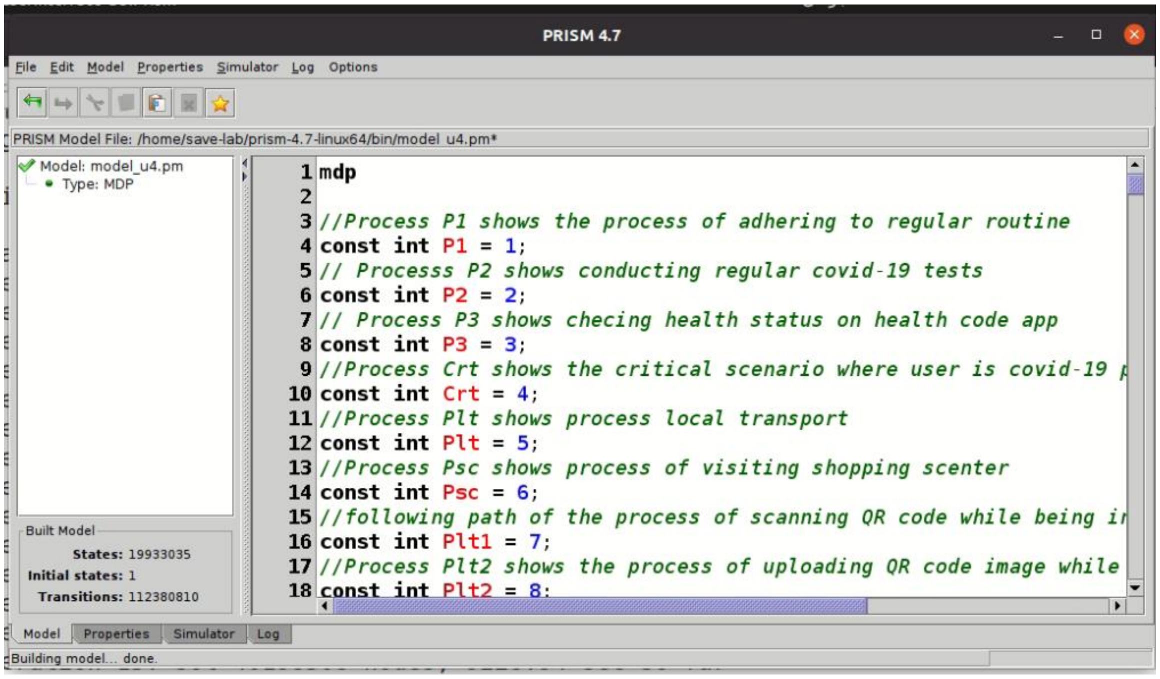
The PRISM is a powerful tool in the field of formal verification, it has its own system description language. Inspired by the Reactive Modules formalism, this language provides an elegant and intuitive means of specifying few fundamental model types: Discrete-Time Markov Chains (DTMCs), Continuous-Time Markov Chains (CTMCs) and Markov Decision Processes (MDPs).

PRISM has two essential components, Modules and Variables. Modules are building blocks encapsulate functionality. Each module contains localized variables, often integers, with well-defined roles. Think of them as the system's actors, each contributing to the grand narrative. Whereas PRISM variables serve as the threads connecting modules. Their valuations—snapshots of the system's state—reveal the bigger picture. Transitions occur when local variables change within a module, and the entire system evolves as modules interact.

Whether asynchronous, where modules transition independently, or synchronous, where they move in dependence, PRISM manages the division of probabilities. As we explore the Chinese health code system, we glimpse its inner workings—one transition at a time.

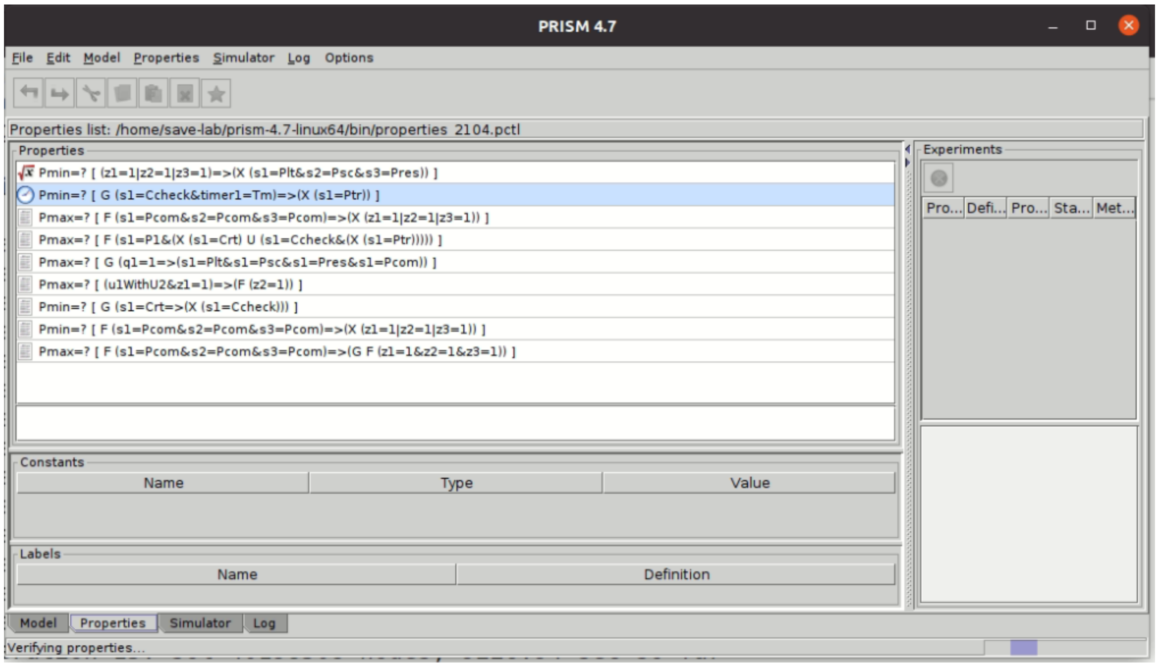
The following are some screenshots of PRISM Model Checker. Figure 3.3 shows the modeling area where users create models using the modeling language of PRISM. Figure

3.4 shows the properties writing area, here users define properties for the created model. Figure 3.5 shows the simulation window, here the behavioral testing of the model is done.


PRISM 4.7
File Edit Model Properties Simulator Log Options
PRISM Model File: /home/save-lab/prism-4.7-linux64/bin/model_u4.pm*
Model: model_u4.pm
Type: MDP
Built Model
States: 19933035
Initial states: 1
Transitions: 112380810
Model Properties Simulator Log
Building model... done.

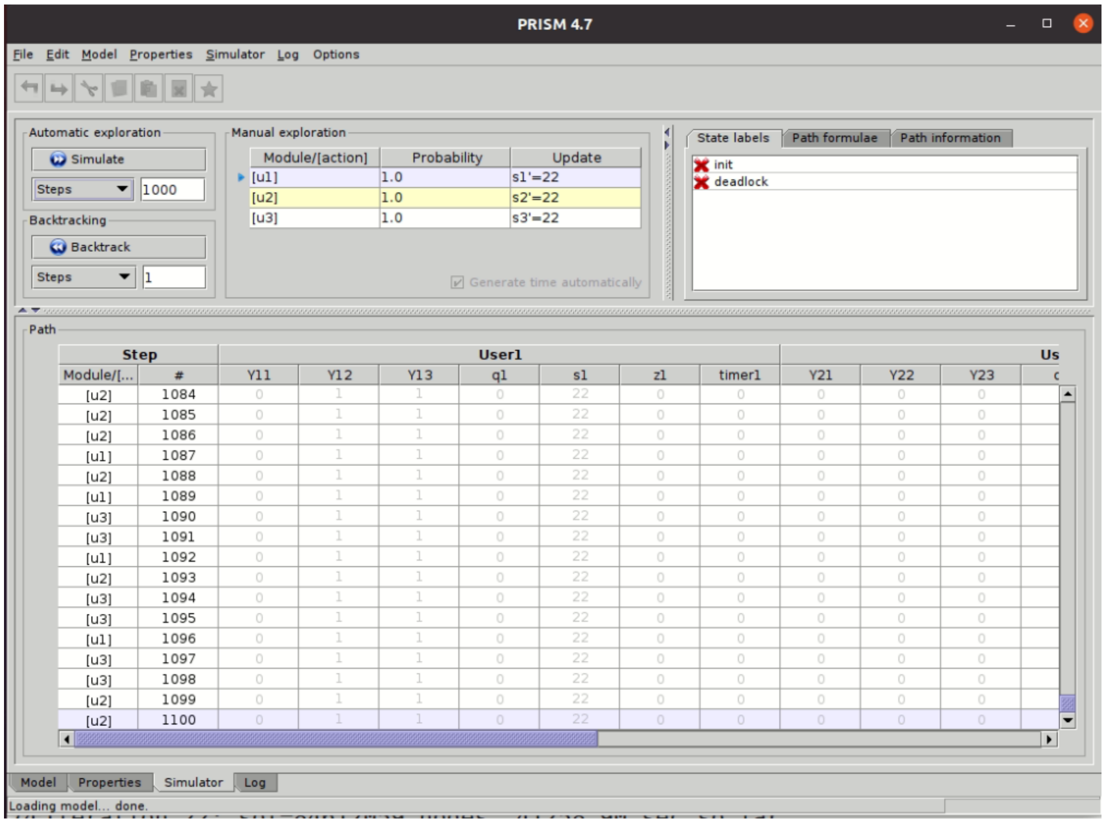
```
mdp

//Process P1 shows the process of adhering to regular routine
const int P1 = 1;
// Process P2 shows conducting regular covid-19 tests
const int P2 = 2;
// Process P3 shows checing health status on health code app
const int P3 = 3;
//Process Crt shows the critical scenario where user is covid-19 p
const int Crt = 4;
//Process Plt shows process local transport
const int Plt = 5;
//Process Psc shows process of visiting shopping scenter
const int Psc = 6;
//following path of the process of scanning QR code while being in
const int Plt1 = 7;
//Process Plt2 shows the process of uploading QR code image while
const int Plt2 = 8;
```

Figure 3.3: Modeling window of PRISM Model Checker


PRISM 4.7
File Edit Model Properties Simulator Log Options
Properties list: /home/save-lab/prism-4.7-linux64/bin/properties_2104.pctl
Properties
Experiments
Pro... Defi... Pro... Sta... Met...
Constants
Name Type Value
Labels
Name Definition
Model Properties Simulator Log
Verifying properties...

```
Pmin=? [ (z1=1|z2=1|z3=1)=>(X (s1=Plt&s2=Psc&s3=Pres)) ]
Pmin=? [ G (s1=Ccheck&timer1=Tm)=>(X (s1=Ptr)) ]
Pmax=? [ F (s1=Pcom&s2=Pcom&s3=Pcom)=>(X (z1=1|z2=1|z3=1)) ]
Pmax=? [ F (s1=P1&(X (s1=Crt) U (s1=Ccheck&(X (s1=Ptr))))) ]
Pmax=? [ G (q1=1=>(s1=Plt&s1=Psc&s1=Pres&s1=Pcom)) ]
Pmax=? [ (u1WithU2&z1=1)=>(F (z2=1)) ]
Pmin=? [ G (s1=Crt=>(X (s1=Ccheck))) ]
Pmin=? [ F (s1=Pcom&s2=Pcom&s3=Pcom)=>(X (z1=1|z2=1|z3=1)) ]
Pmax=? [ F (s1=Pcom&s2=Pcom&s3=Pcom)=>(G F (z1=1&z2=1&z3=1)) ]
```

Figure 3.4: Properties window of PRISM Model Checker



Module/[action]	Probability	Update
[u1]	1.0	s1'=22
[u2]	1.0	s2'=22
[u3]	1.0	s3'=22

Step		User1										Us
Module/[...	#	Y11	Y12	Y13	q1	s1	z1	timer1	Y21	Y22	Y23	c
[u2]	1084	0	1	1	0	22	0	0	0	0	0	
[u2]	1085	0	1	1	0	22	0	0	0	0	0	
[u2]	1086	0	1	1	0	22	0	0	0	0	0	
[u1]	1087	0	1	1	0	22	0	0	0	0	0	
[u2]	1088	0	1	1	0	22	0	0	0	0	0	
[u1]	1089	0	1	1	0	22	0	0	0	0	0	
[u3]	1090	0	1	1	0	22	0	0	0	0	0	
[u3]	1091	0	1	1	0	22	0	0	0	0	0	
[u1]	1092	0	1	1	0	22	0	0	0	0	0	
[u2]	1093	0	1	1	0	22	0	0	0	0	0	
[u3]	1094	0	1	1	0	22	0	0	0	0	0	
[u3]	1095	0	1	1	0	22	0	0	0	0	0	
[u1]	1096	0	1	1	0	22	0	0	0	0	0	
[u3]	1097	0	1	1	0	22	0	0	0	0	0	
[u3]	1098	0	1	1	0	22	0	0	0	0	0	
[u2]	1099	0	1	1	0	22	0	0	0	0	0	
[u2]	1100	0	1	1	0	22	0	0	0	0	0	

Figure 3.5: Simulation window of PRISM Model Checker

The capacity of a module to move from one state to another, or the transitions that are permissible in a given state are identified by a set of commands Some of the system dependent features of the language are as follows – Each command contains a guard that determines its identity, a sub portion of the global state space and one or more a value, each of which corresponds to an example of how the same course module might transform. Conceptually, in general, it seems quite reasonable, if the model is in a state resulting in the guard of a command then the module can make the transitions defined by the updates of that command. The likelihood of each transition being made is also specification individual transition probabilities by the command. The nature of this information is extremely specific and depends on the type of model being used. This will be made clearer in the subsequent sections by means of several small-yet-significant examples.

Pseudocode: A Generic module pseudocode

```
module User_n
        S_n ← [1 … 22] ;
        Z_n ← [0 … . 1 ] ;
        Timer_n ← [0 … . 7];
        $[u_n]\ \ s_n = P_1 \ \rightarrow 1 : (S'_n = \ P_2);$
        $[u_n]\ \ s_n = P_2 \ \rightarrow 1 : (S'_n = \ P_3);$
        $[u_n]\ \ s_n = P_3 \ \rightarrow x : (S'_n = \ C_{rt})\ \&\ (Z_n = \ 0) + \ \ 1 - \ x : (S'_n = \ P_{tr});$
        if ( $(U_n With U_2)$ =1) & $(Z_2$ =1)) | ( $(U_n With U_3)$ =1) & $(Z_3$ =1)
            $q'_n = 1$ ;
        endif
        if ( $(S_n$ =1) | $(q_n$ =1)) & $(Timer_1 < Tm)$
            $1 : (S'_n = \ C_{check})\ \&\ (Z'_n = \ 1)\ \&\ \ (q'_4 = \ 1)\&\ \ (Timer'_n = Timer_n + 1);$
        endif
        if $(S_n = C_{check})\ \ \&\ \ (Timer_1 = Tm)$
            $1 : (S'_n = \ P_{tr})\ \&\ \left(q'_n = \ 0\right)\ \&\ \ (Timer'_n = 0);$
        endif
        repeat
            $[u_n]\ \ (\ s_n = P_{tr})\ \&\ \left(q'_n = \ 0\right) \rightarrow 0.25 : (S'_n = \ P_{lt})\ +$
                      $0.25 : (S'_n = \ P_{sc})\ +$
                  $0.25 : (S'_n = \ P_{res})\ +\ 0.25 : (S'_n = \ P_{com})$ ;

            $[u_n]\ \ (\ s_n = P_{lt}) \rightarrow 1 - y : (S'_n = \ P_{lt1})\ +\ y : (S'_n = \ P_{lt2});$
            $[u_n]\ \ (\ s_n = P_{lt1}) \rightarrow 1 : (S'_n = \ P_{tr})\ \&\ \ (Y'_{41} = \ 1);$
```

```
24:                 [$u_n$] ( $s_n = P_{lt2}$) $\rightarrow 1: (S'_n = P_{tr})$;
25:                 [$u_n$] ( $s_n = P_{com}$)
                            $\rightarrow x: (S'_n = P_{home})$ & $(Z'_n = 1)$
26:                         $+ x: (S'_n = P_{home})$;
                    [$u_n$] ( $s_n = P_{home}$) $\rightarrow 1: (S'_n = P_{home})$;
            until $s_n = P_{home}$
        endmodule
```

Table 3.1: The Pseudocode for PRISM model

3.4 The Chinese Health-Code App for Quarantine Management

The Chinese health code app[4], known as the Green Health Code (Chinese: 健康码, Jiànkāngmǎ), was widely used during the COVID-19 pandemic in mainland China.

The Health Code served as an e-passport that reported an individual's risk level based on their travel history, residence, and medical records. It helped manage quarantine measures and track potential exposure to infection. After providing relevant information, the app generated a QR code with one of three colors: Green: Allowed unrestricted movement. Yellow or Red: Indicated the need to report to authorities or quarantine.

By April 2020, over 200 cities and 20 provinces employed health codes supported by Alipay (a digital platform). Different localities had their own versions of health codes. Despite assumptions that health codes would phase out post-pandemic, research suggests their continued use beyond the health crisis.

[4] Health Code: What and how? | govt.chinadaily.com.cn

The Health Code ran through platforms like Alipay and WeChat as shown in Figure 3.6, requiring real-name registration and personal data related to travel history and health records[5]. It played a crucial role in managing COVID-19 risks and ensuring public safety.

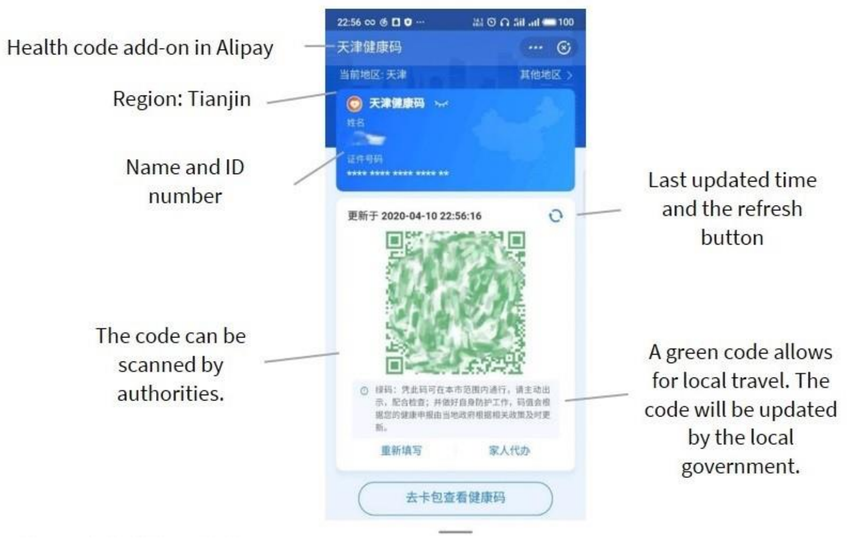

Health code add-on in Alipay
Region: Tianjin
Name and ID number
Last updated time and the refresh button
The code can be scanned by authorities.
A green code allows for local travel. The code will be updated by the local government.
天津健康码
当前地区: 天津
其他地区
更新于 2020-04-10 22:56:16
重新填写
家人代办
去卡包查看健康码

Figure 3.6: Screenshot of Health code in Alipay[6]

[5] China's Health Code app showcases the extreme smart surveillance regime | by Yuki Yuen | Assay | Medium
[6] China's Novel Health Tracker: Green on Public Health, Red on Data Surveillance | Trustee China Hand | CSIS

CHAPTER 4: PROPOSED METHODOLGY

Every Chinese health code system should meet generic requirements for proper/complete operation. Its architecture should be comprised of the following, it should be generic, scalable, and be able to cope with faults. In terms of safety, it should not allow the health code to be issued or used outside the specified parameters. The architecture should be generic and scalable, accommodating various use cases and scenarios. It must be fault-tolerant, handling unexpected situations or errors effectively. The health code system must operate within a specified range to ensure accurate health status representation.

4.1 Modeling Chinese Health-Code App in PRISM

Our Proposed methodology will help us verify the Chinese Health-Code App that can possibly behave unexpectedly or can be bypassed by user who's contract tracing path is not valid for certain paths. The Health Code, as stated above, is an ever-evolving code that is developed in the context of a mobile-app.

It consists of three colors green, red, and yellow. From the disclosed data, users input information by passing through the application, their travel history, their domicile and their medical history. This self-declared data then is integrated with disease control related big data. From the data gathered, the program creates codes in the form of QR code for each person. The QR code reflects the user's risk level: The QR code reflects the user's risk level: **Green:** Certainly, one can move from one place to another safely without being harassed or attacked by the inhabitants of the area. **Yellow:** They have had possible exposure or are at a low risk of infection. **Red:** High risk (for example contact with people who are infected).

The Green Health Code acts as a permit to move between places such as public transport, working places, and the supermarkets. Ideas may include if a user gets close to an infected person, a certain code on the screen could change color, say red, and the user would be cautioned to self-isolate.

4.2 Proposed Methodology

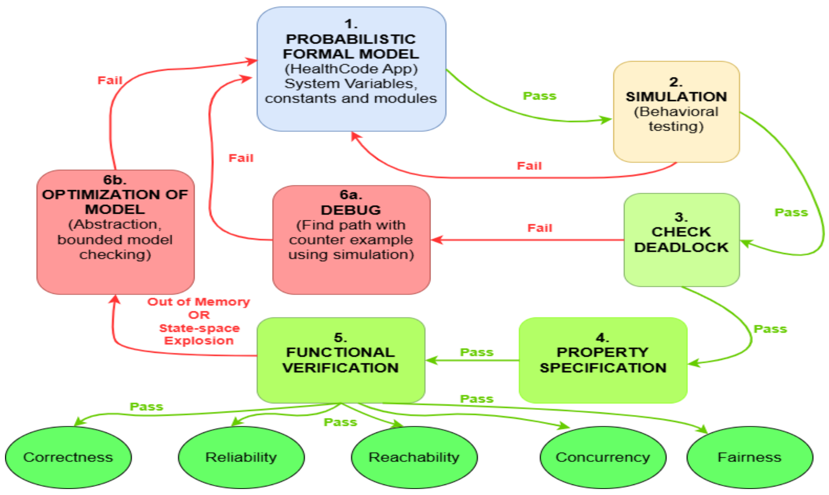

1. PROBABILISTIC FORMAL MODEL (HealthCode App) System Variables, constants and modules
2. SIMULATION (Behavioral testing)
3. CHECK DEADLOCK
4. PROPERTY SPECIFICATION
5. FUNCTIONAL VERIFICATION
6a. DEBUG (Find path with counter example using simulation)
6b. OPTIMIZATION OF MODEL (Abstraction, bounded model checking)
Pass
Fail
Out of Memory OR State-space Explosion
Correctness
Reliability
Reachability
Concurrency
Fairness

Figure 4.1: Proposed Methodology

4.2.1 Probabilistic Formal Model

The first step in the proposed methodology is, depicted in Figure 4.1, for formal verification of Chinese Health Code system is to develop a formal model for its system behavior. Here we have taken an idea from the proposed methodology of Iram Bhatti et. al[33], they used the same modeling tool and verification techniques in their work. Now for our work selecting a suitable model for the modeling is one of the most important tasks. In our case we propose to use MDP as the nature of system is non-deterministic. The proposed modeling is based on the modular approach. Each user and its working are combined in a single module, the number of modules is equal to the number of users in this system.

Components of a Probabilistic Formal Model

The following steps will be used to model the Chinese Health-Code system in PRISM:

4.2.1.1 Identifying Modules

In the context of the Chinese quarantine management system, the first step of modeling involves identifying the modules within the system. Each module corresponds to a specific functionality, such as health status management, keeping a trace of contact, quarantine management, and User Management. These modules are then implemented as Finite-State Machines (FSMs) with certain probabilities. The behavior of each block within a module is expressed using a Markov Chain Model. This approach ensures efficient and reliable management of quarantine protocols, contributing to effective pandemic control.

4.2.1.2 Model Construction

For this, one should use the PRISM language and create modules reflecting various aspects of the system. Enumerate and assign variables that suit the various states a user should be going through such as ‘healthy’, ‘infected’ or ‘quarantined’. Define transitions from one state to another with connected probabilities, these can be derived from possible statistical data or assumption.

4.2.1.3 Setting constants for States

Every user within this system has a state to reside in, a user starting from a state must traverse these states. Here, the concept is to map every state on a constant (constant variables) and make it as a representation of state.

4.2.1.4 Identifying Inputs/Outputs (Variables)

Sharing the trace of different modules is very important and crucial, in our case each module represents the user, and their actions are controlled by variables. In the Control Algorithm, data sharing occurs between different modules through variables created within each module. These variables act as global variables and can be accessed from any module. Additionally, these variables are defined with upper and lower limits and support common data types.

4.2.1.5 Initialization

In the context of a control algorithm, data sharing occurs among various modules through the use of variables. Each module initializes its own set of variables, which act as global entities accessible from any other module. These variables hold data relevant to the module's specific tasks, such as quarantine, positivity, or control parameters. For example, a user control module might have variables like timer and positivity. These variables are defined with upper and lower limits and support common data types (such as integers, floats, or booleans). Importantly, PRISM executes these modules in parallel, allowing for efficient processing. Additionally, you can create multiple instances of the same module by changing the variable names, ensuring flexibility and scalability within the system.

Component Name	***Count***
Module Type	MDP
Total Modules	3
Number of Constants for States representation	21
Number of Constants for probability distribution	2
Number of Formulas	3
Number of variables in each module	7
Number of global variables	1

Table 4.1: Table containing some insights of the model to be created in PRISM

4.2.2 Simulation

After modeling is done in PRISM, it undergoes compilation to identify any errors while implementation. The compilation process also assesses whether any state probabilities are not realistic. Subsequently, the model should be evaluated using PRISM's random and interactive simulator. This simulation, which employs random test vectors, often reveals invalid functioning. These issues can then be addressed in the model. The primary purpose of the model simulation is to identify flaws that has been made during implementation, before undertaking comprehensive and relatively time-consuming formal verification.

Furthermore, after simulating the model, it’s essential to analyze the results. This analysis involves examining the behavior of the system under different scenarios, checking for unexpected outcomes, and ensuring that the model aligns with the intended specifications. Additionally, any discrepancies or discrepancies between the simulated behavior and the expected behavior should be thoroughly investigated and resolved.

4.2.3 Deadlock Freedom

Ensuring the absence of deadlocks in a system model constitutes a fundamental verification step. By conducting this check, we identify states or scenarios where users are unable to take further actions. This property serves as a safeguard, guaranteeing that our implemented model remains free from programming flaws. Consequently, this verification is valuable across a wide range of applications.

4.2.4 Optimization

Model optimization involves simplifying a complex model without losing crucial information. This is crucial for managing the state space explosion problem. Key parameters for abstraction in this step include the number of places user visit, number of days a user is quarantined and iteration of user activities. By abstracting these elements, the model becomes more manageable while preserving essential characteristics.

4.2.5 Out of Boundary Operation

The paramount consideration for any system is to remain within its designated operational boundaries at all times. If it ventures beyond these boundaries, it can lead to unintended operations and disrupt the expected processes. To prevent this, the system undergoes rigorous checks for boundary crossovers, assessing the likelihood of such occurrences. These probabilities are then analysed to understand and mitigate potential crossover scenarios.

4.2.6 Property Specification and Functional Verification

The next phase involves defining and evaluating the performance metrics of the Chinese health code system. Key areas of focus include the system's effectiveness in preventing disease spread, fairness in data handling, the ability to handle simultaneous user interactions, the accessibility of places within the system, and the accuracy of health status determination. These properties are formally defined and analyzed to ensure the system's reliability. If performance issues arise, the system's data structure or processing logic is optimized. In case of errors, system logs are analyzed to identify and rectify problems.

CHAPTER 5: MODEL THE CHINESE HEALTH-CODE APP

In this section we will discuss about the formal modeling of potential model of Chinese Health-code system, as mentioned in figure 5.1.

5.1 Potential Flowchart diagram of Chinese Health-code App

This is the first step in this modeling process, here each state represents some actions made by the user. This is the flow for each user. The second step after the flow diagram conceptual modeling, as shown in figure 4.1, is to develop the model for Chinese Health-code Algorithm.

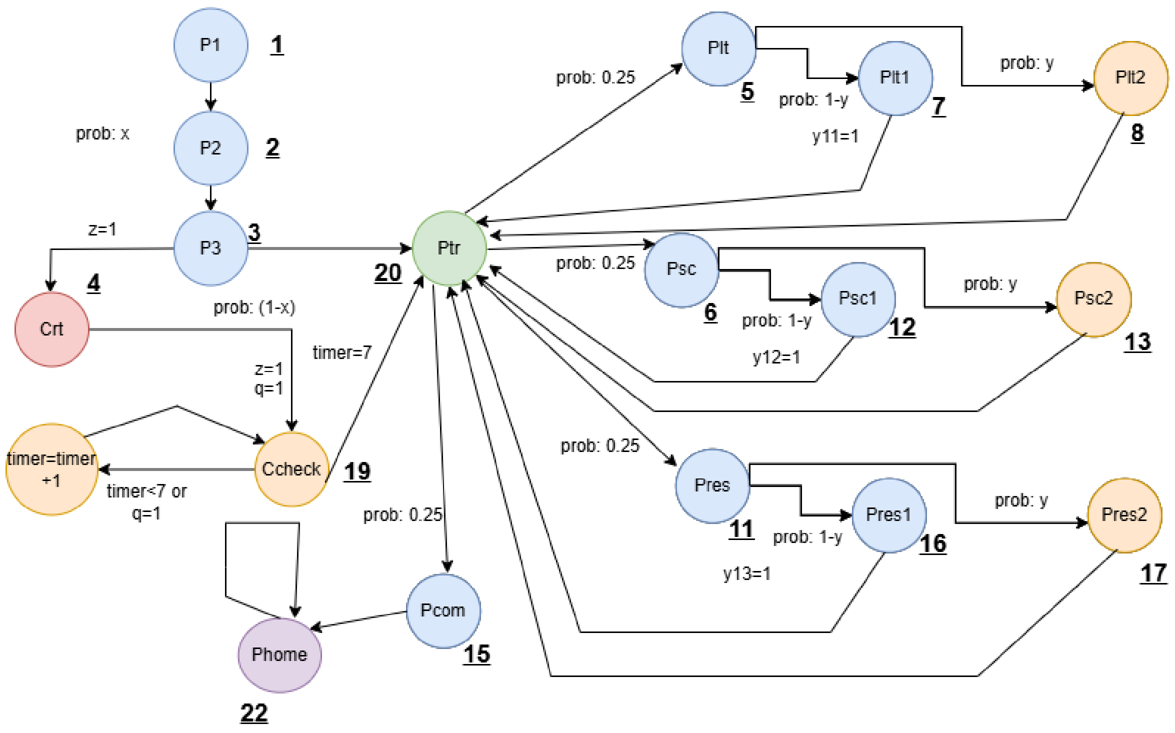


Figure 5.1: Potential Flowchart of Chinese Health-code App

Here, the most appropriate model type for our system is MDP, so we opted this model type for our model. Here, a user starts his journey in this system by registering on this app and integrating his health condition in App.

Let’s discuss this model and its states. Each state represents a scenario, we have used constants instead of numeric values for simplicity as shown in Table 5.1. This is a case study and we have picked some limited scenarios like local transport, shopping center, restaurant and community. Here, a trace of each user is kept who visits any of these places. We will discuss each part of this system in parts in the following sections.

Sr. No	***Abbreviation***	***Meaning***
1	$s_1, s_2, \ldots$	State of user 1 and respectively user 2 etc
2	P_1	Process P1 shows the process of adhering to regular routine
3	P_2	Process P2 shows conducting regular covid-19 tests
4	P_3	Process P3 shows checking health status on health code app
5	C_{rt}	Process Crt shows the critical scenario where user is covid-19 positive
6	P_{lt}	Process Plt shows process local transport
7	P_{sc}	Process Psc shows process of visiting shopping center
8	P_{sc1}	following path of the process of scanning QR code while entering in the shopping center
9	P_{sc2}	Process Plt2 shows the process of uploading QR code image while being in the shopping center
10	P_{lt2}	Process Plt2 shows the process of uploading QR code image while being in of local transport
11	P_{res}	Process Pres shows the process of entering the restaurant.
12	P_{res1}	Pres1 shows the process of entering the restaurant scenario
13	P_{res2}	Pres2 shows the Process of uploading QR code gallery image while being in the restaurant
14	C_{check}	Process for COVID-19 test
15	P_{tr}	Process for general entry into any of the scenarios
16	P_{home}	Process of reaching home safely
17	T_m	Maximum timer limit
18	$U_1 \cup U_2$	For syncing contact tracing of user 1 with user 2
19	$U_1 \cup U_3$	For syncing contact tracing of user 1 with user 3
20	$timer_1$, $timer_2$	timer for quarantine of user 1, user 2 and so on
21	Y_{11}	Trace of User 1 in local transport
22	Y_{12}	Trace of User 1 in Shopping center
23	Y_{13}	Trace of User 1 in restaurant
24	z_1	Health status of user 1 (z1 means user1 is COVID-19 positive)
25	u_1	For keeping few states of user1 synchronous
26	P_{lt1}	following path of the process of scanning QR code while being in of local transport
27	x	Positivity Rate
28	y	Fraud rate

Table 5.1: Table containing the details of constants, variables and formulas of model

5.2 Model details and Pseudocode of different functionalities

The system developed in PRISM contains functions in each module. Each module is named as User and each module contains same set of functions like, handling of success scenarios, critical paths, quarantine related transitions, handling valid and invalid path for each scenario e.g visiting local transport, shopping center and restaurant. In the following section we will discuss the functions and its pseudocode.

5.2.1 Variables

- **S_1:** Represents the current state of User 1 (P1, P2, ..., Phome) - This is the core variable defining User 1's progress through the model.
- **Z_1:** Indicates user's health status (0 - Healthy, 1 - Positive) - Tracks if the user is infected.
- **$timer_1$:** Timer for quarantine period (0 to Tm)[7] - Keeps track of the quarantine duration.

5.2.2 Flags

- **$U_1 \cup U_2$:** Flag indicating contact with infected User 2 (True/False) - Used for contact tracing.
- **$U_1 \cup U_3$**: Flag indicating contact with infected User 3 (True/False) - Used for contact tracing.
- **Y_{11}, Y_{12}, Y_{13}:** Flags set during local transport, shopping, or restaurant visit (0 or 1) - Track specific activity flags.

5.2.3 Initial State:

[7] Isolation and Precautions for People with COVID-19 | CDC

‘init’ keyword is used to assign initial values to variables.

- **$S_1 = P_1$**: User starts in the initial state P1.
- **$Z_1 = 0$**: User is healthy initially.
- **$timer_1$ = 0**: Quarantine timer starts at 0.

5.2.4 Transitions

Each transition represents a possible change in state based on conditions and probabilities. Defined using square brackets [] and a label u_1,u_2,u_3 respectively, based on the module.

- **The general format is**: [Guard] -> [Probability]: (Next State) & (Variable Updates).
- **Guard**: A Boolean expression that determines if the transition can occur (e.g., current state, flag values).
- **Probability**: A numerical value between 0 and 1 indicating the likelihood of the transition happening (not always present).
- **Next State**: The state the module moves to if the transition fires.
- **Variable Updates**: Updates to other variables within the module that occur along with the state change.

5.2.5 Functionality of Starting States/ A Pseudocode

This part of code simply making transition with probability equals to 1 in first two states i.e P_1 to P_3, there is no guard on these states. Furthermore, when it reaches state P3, it then go to state C_{rt} with probability ‘x’ or goes to state P_{tr} with probability ‘1-x’

Pseudocode: Starting States

```
if $s_n = P_1$
        $1 : (S'_n = P_2);$

endif

if $s_n = P_2$
        $1 : (S'_n = P_3);$

endif

if $s_n = P_3$
        $[u_n]\ s_n = P_3 \rightarrow x : (S'_n = C_{rt}) \& (Z_n = 0) + 1 - x : (S'_n = P_{tr});$

endif
```

5.2.6 Functionality of Critical Path/ A Pseudocode

In this part of module, the colour code of user is set ‘red’ and user is quarantined for a specific time. These all operations are managed through variables z, q and timer, each representing health status, quarantine status and timer for the quarantine. If the user completes the maximum timer limit he/she is set to default value, he/she is allowed to enter any scenario without any restriction.

Pseudocode: The Critical Path

```
if $(S'_n = C_{rt}) | (Z_n = 1)$
        $increment\ the\ timer\ by\ 1\ day$ ;
        $1 : (S'_n = C_{check}) \& (Z'_n = 1)$
endif
```

5.2.7 Functionality of Checking Quarantine Status/ A Pseudocode

It is regularly checked for a user if he/she is in quarantine. In such cases if the user completes the maximum timer limit, he/she is set to normal user, assigned with green code and now they are allowed to enter the normal routine.

Pseudocode: The Critical Path

```
1:  if $(S'_n = C_{check})$ & $(q_n = Tm)$
2:          $1 : (S'_n = P_{tr})$ & $(q'_n = 0)$ & $(Timer'_n = 0)$ & $(Z'_n = 0)$;
3:  endif
```

5.2.8 Functionality of Normal Routine

A user is allowed to enter a normal routine if he/she clears some checks, like colour code is green and he/she has not come in contact with Covid-19 positive user.

Pseudocode: Normal Routine

```
1:  if $(s_n = P_{tr})$ & $(q'_n = 0)$
2:          $0.25 : (S'_n = P_{lt}) + 0.25 : (S'_n = P_{sc}) + 0.25 : (S'_n = P_{res})$
3:                  $+ 0.25 : (S'_n = P_{com})$ ;
4:  endif
```

5.2.9 Functionality of Entering Local transport/ Shopping Centre/ Restaurant

As all the scenarios have the same mechanism i.e. local transport, shopping center and restaurant, so will cover a single scenario. In this scenario a user is opting for local transport, there are two options. Firstly he/she can either use a live camera to scan the QR code for which it will take the data of user and allow/disallow him/her from entering and record his/her entry. Secondly, a user can upload an image of QR code from gallery, which will record the entry against the QR code which has been scanned.

Pseudocode: Entering Local transport/ Shopping Centre/ Restaurant

```
1:  if ( $s_n = P_{lt}$)
2:          $1 - y: (S'_n = P_{lt1}) + y: (S'_n = P_{lt2});$
3:          if ( $s_n = P_{lt1}$)
4:                  $1: (S'_n = P_{tr})$ & $(Y'_{41} = 1);$
5:          if ( $s_n = P_{lt2}$)
6:                  $1: (S'_n = P_{tr});$
7:
8:              $0.25 : (S'_n = P_{lt}) + 0.25 : (S'_n = P_{sc}) + 0.25 : (S'_n = P_{res})$
9:                                  $+ 0.25 : (S'_n = P_{com})$ ;
```

5.2.10 Community to Home

A user entering a community after visiting different scenarios/places will eventually reach his/her home. Here, a user will remain stay at home with no further transitions. There is a possibility that a user who has visited any or all places will get Covid-19 positive at any time, the probability for his/ her possible infection is set to ‘x’.

Pseudocode: Community to Home

```
1:  if ( $s_n = P_{com}$)
2:          $x: (S'_n = P_{home})$ & $(Z'_n = 1) + x: (S'_n = P_{home});$
3:  elseif
```

CHAPTER 6: RESULTS AND DISCUSSION

In this chapter, we rigorously verify the formal model of the Chinese health code system. To achieve this, we employ property specifications outlined in our proposed methodology. For verification purposes we have set two variables positivity rate 'x' and fraud rate 'y', now we have used a set of values for them both i.e. started value of x and y from 0, 0 then 0, 0.2 and so on till 1, 1. Now their combination gave us 36 different combinations.

So, our experimental setup contains probabilities for 36 combinations. This helps us understand the pattern and behavior of property against different combinations of probabilities. The below tables show the specifications of system we have used for our experiments, shown in Table 6.1 and the second one, Table 6.2 shows some insights of our experimental setup.

System Specs	*Value*
OS	Ubuntu 20.04.4 LTS x86_64
Host	PowerEdge T320
Kernel	5.15.0-113-generic
Shell	bash 5.0.17
Resolution	1600x900
Terminal	gnome-terminal
CPU	Intel Xeon E5-2407 v2 (4) @ 2.4
GPU	6:00.0 Matrox Electronics Syst
Memory	1072MiB / 32017MiB

Table 6.1: The details of Computer System used for thesis

Constants/ Variables	*Count*
Number of Users	3
Positivity Rate (Starting point)	0
Positivity Rate (End point)	1
Jump	0.2
Fraud Rate (Starting point)	0
Fraud Rate (End point)	1
Jump	0.2
Total Experiments for each property	36

Table 6.2: Experimental setup for Property verification

6.1 Deadlock Freedom

In the context of Chinese health-code modules, a crucial requirement is that they avoid getting stuck in any particular state. To verify this property, we focus on deadlock freedom—the assurance that the underlying control algorithm never leads to a deadlock situation.

E [F “deadlock”]

We leveraged the PRISM model checker, specifically its built-in deadlock property verification. This property examines whether, for certain states, transitioning from the present state to a future state would result in a deadlock. Our findings were reassuring: the algorithm demonstrated deadlock freedom. The failing property check confirmed that no deadlocks exist within the model.

6.2 Reachability

Beyond ensuring deadlock freedom, another essential characteristic for any user within a system is the ability to reach desired states while adhering to valid and permissible conditions. Notably, the movement of users across different areas within a society, following valid paths, is of utmost importance. Consequently, verifying the reachability property becomes even more critical.

Specifically, we must address scenarios where the algorithm restricts a user's movement—especially during quarantine or, more critically, when they test positive. The reachability property can be rigorously verified by checking whether, when a user is quarantined for a specified duration, they can move from their initial location to the required destination within a finite number of steps. Our verification process involves associating this property with the system's behavior.

6.2.1 Reachability of all Users in a State

Max probability that in future all uses are visiting restaurant and any of them is following the alternate path and getting positive

Property:

"$P_{max} = ?\ [\ F\ ((s_1 = P_{res}\ \&\ s_2 = P_{res}\ \&\ s_3 = P_{res}\ )\ \&\ (s_1 = P_{res2}\ \&\ s_2 = P_{res2}\ \&\ s_3 = P_{res2}\ )\&\ (F\ (z_1 = 1\ |\ z_2 = 1\ |\ z_3 = 1))\ ]$"

Empirical Data:

		X (Positivity Rate)					
		0	0.2	0.4	0.6	0.8	1
Y (Fraud Rate)	0	0	0	0	0	0	0
	0.2	0	0.0968	0.2149	0.3110	0.3580	0.3666
	0.4	0	0.0868	0.1830	0.2596	0.2978	0.3053
	0.6	0	0.0759	0.1476	0.2015	0.2286	0.2344
	0.8	0	0.0657	0.1138	0.1446	0.1591	0.1622
	1	0	0.0610	0.0980	0.1170	0.1240	0.1250

Table 6.3: Data of Reachability Property 1

Graphical Representation:

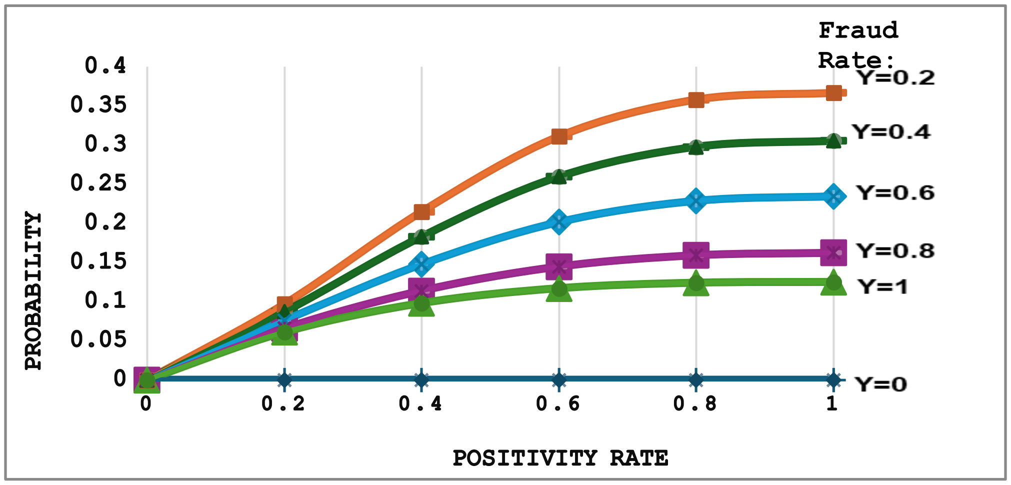


Figure 6.1: Graphical Representation of Reachability Property 1

Discussion on results:

The probability values show an increasing trend for each values of Y (0.2 to 1) as X increases from 0.0 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0.2 to Y=1.0 for values of X (0.2 to 1) . The results shows that fraud rate will affect the reachability of users to valid paths, as he/she will restrict the possible valid paths.

6.2.2 Ensuring the Contract Tracing

It calculates the maximum probability (P_{max}=?) of a path where the model eventually reaches a state with:

- All flags (Y_{11}, Y_{21}, Y_{31}) set to 1, indicating potential contact with infected users 1, 2, and 3.
- $User_1$ being healthy initially (z_1=0).

However, reaching this state doesn't guarantee that $User_1$ will eventually become infected (F (z_1=1)) because the second F operator is independent. Infection could happen later through other paths.

Property:

$$``P_{max} = ?\,[\,F\,((\,Y_{11} = 1\ \&\,Y_{21} = 1\,\&\ Y_{31} = 1\,\&\ Z_1 = 0\,)\,\&\,(F\,Z_1 = 1)\,)\,]"$$

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0	0.1988	0.3220	0.3876	0.4137	0.4184
	0.2	0	0.1633	0.2653	0.3204	0.3429	0.3473
	0.4	0	0.1202	0.1964	0.2384	0.2564	0.2601
	0.6	0	0.0700	0.1155	0.1415	0.1533	0.1561
	0.8	0	0.0206	0.0345	0.0430	0.0473	0.0485
	1	0	0	0	0	0	0

Table 6.4: Data of Reachability Property 2

Graphical Representation:

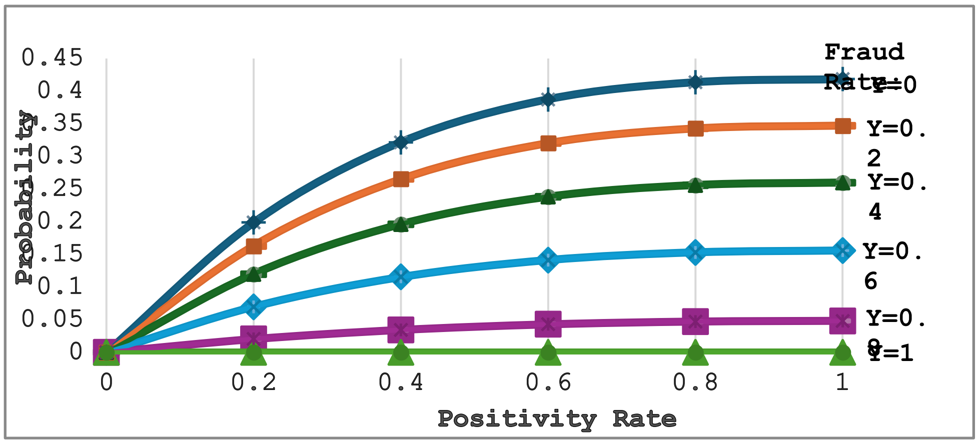


Figure 6.2: Graphical Representation of Reachability Property 2

Discussion on results:

The probability values show an increasing trend for each values of Y (0.0 to 0.8) as X increases from 0.2 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0.2 to Y=1.0 for values of X (0.2 to 1). The result is making it clear that maximum fraud rate will affect the system and an infected user will not be traced due to following invalid path.

6.3 Correctness of System

The correctness of the system is one of the most crucial parts. This is checked through different ways, lets discuss few of them.

The transitions between states within module User1 should accurately reflect the intended behavior based on conditions (e.g., transitioning to quarantine upon testing positive). Changes to variables like health status and flags (infected user contact) should be consistent with the state transitions and model logic. The model should correctly represent how users come in contact with each other and how that contact is reflected in the infected user flags.

The quarantine logic (entering C_{check} state, timer updates) should function the same way for all users regardless of their specific path through the model. The model shouldn't have situations where all users get stuck in a state with no possibility of progressing further.

The model shouldn't have situations where users keep repeatedly entering and exiting specific states without making meaningful progress (e.g., endlessly looping in the local transport states). The combined behavior of all user modules and the environment should achieve the intended purpose of the system being modeled. For example, if the goal is to simulate disease spread, the model should accurately depict user interactions and infection probabilities.

6.3.1 Property 1

$$“P_{max} = ?\,[\,F((\,Y_{11} = 0\ \& Y_{12} = 0\,\&\ Y_{13} = 0\,\&\ Z_1 = 1\ )\&\,(\,q_1 = 1))]”$$

Description: This property focuses on the correctness of individual user behavior (User1) and can be seen as a violation of the expected behavior. It calculates the maximum probability (P_{max}=?) of a path existing in the model where eventually (F) all the following conditions hold true for $User_1$:

- Y_{11}=0: User$_1$ did not visit a local transport location (flag not set).
- Y_{12}=0: User$_1$ did not go shopping (flag not set).
- Y_{13}=0: User$_1$ did not visit a restaurant (flag not set).
- z_1=1: User$_1$ is infected (health status flag).
- q_1=1: User$_1$ is in a quarantine-related state (quarantine flag).

Empirical Data:

		X (Fraud Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0	0.2	0.4	0.6	0.8	1
	0.2	0	0.2	0.4	0.6	0.8	1
	0.4	0	0.2	0.4	0.6	0.8	1
	0.6	0	0.2	0.4	0.6	0.8	1
	0.8	0	0.2	0.4	0.6	0.8	1
	1	0	0.2	0.4	0.6	0.8	1

Table 6.5: Data of Correctness Property 1

Graphical Data:

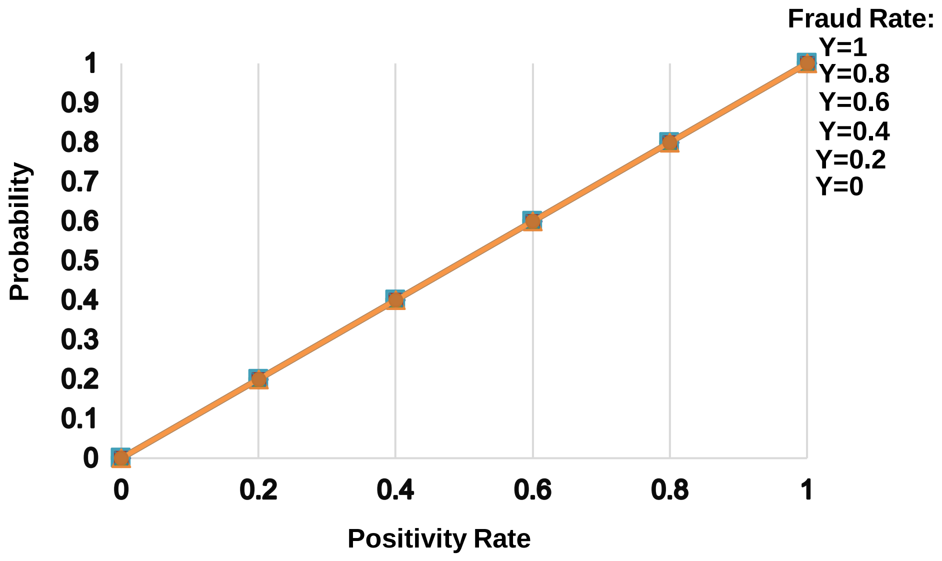


Figure 6.3: Graphical Representation of Correctness Property 1

Discussion on result:

This property indicates a potential correctness issue because:

User1 becomes infected (z_1=1) despite not engaging in any of the potentially risky activities (Y_{11}=0, Y_{12}=0, Y_{13}=0) that would normally set the corresponding flags.

User1 is quarantined (q_1=1), which might not be justified if there's no apparent reason for quarantine based on the listed conditions.

6.3.2 Property 2:

$$“P_{max} = ?\,[\,F\,((s_1 = P_{res}\,\&\,s_2 = P_{res}\,\&\,s_3 = P_{res}\,)\,\&\,(F\,(s_1 = P_{res1}\,\&\,s_2 = P_{res1}\,\&\,s_3 = P_{res1})\,)\,]”$$

Description: This property addresses the correctness of multi-user system interactions, specifically related to restaurant visits. It calculates the maximum probability (Pmax=?) of a path where eventually (F) the following conditions hold true for Users 1, 2 and 3:

- (s_1 = P_{res} & s_2=P_{res} & s3 = P_{res}): All $User_1$, $User_2$ and $User_3$ are simultaneously in the restaurant state (Pres).
- (F (s_1=P_{res1}&s_2=P_{res1}&s_3=P_{res1})): Eventually (F), both users transition to the "restaurant visit complete" state (P_{res1}).

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0.1250	0.2162	0.3092	0.3793	0.4124	0.4184
	0.2	0.0878	0.1526	0.2330	0.3038	0.3404	0.3473
	0.4	0.0527	0.0926	0.1538	0.2170	0.2531	0.2601
	0.6	0.0233	0.0415	0.0768	0.1211	0.1501	0.1561
	0.8	0.0046	0.0081	0.0169	0.0322	0.0454	0.0485
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.6: Data of Correctness Property 2

Graphical Representation:

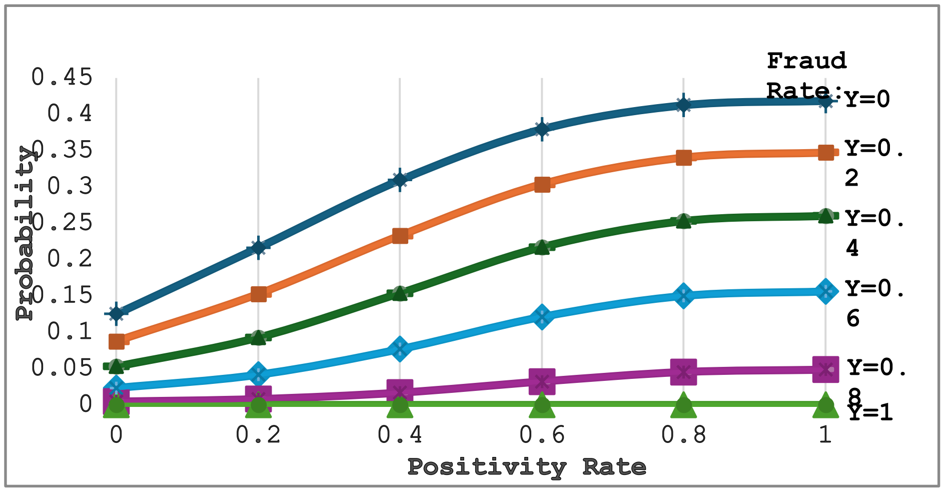


Figure 6.4: Graphical Representation of Correctness Property 2

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 1) as X increases from 0 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0 to Y=1.0 for values of X (0 to 1) . The results shows that with frequent increase in fraud rate the capability of system to make user follow valid path also decreases.

6.4 Fairness in a System

Each user of our system must have to have equal opportunity to visit any place. Like, we might want to ensure all users have a fair chance of eventually visiting a restaurant. This avoids situations where specific users are perpetually blocked from entering the restaurant state due to other users constantly occupying those slots. Properties could be written to reason about the frequency or distribution of restaurant visits across all users.

The spread of infection spread should be balanced, while not strictly fairness, analyzing the overall infection spread across users is relevant. Properties should be there to compare the probability of infection or the time it takes for each user to become infected. This could help identify potential biases in the model regarding infection risk.

Fairness considerations are extended to the entire system, including interactions between all users. Ensuring all infected users have an equal chance of being identified by others. Verifying a similar average quarantine time across users who test positive.

6.4.1 Property 1

This property ensures that the user keeps making progress through the model and doesn't get stuck in an infinite loop within certain states. This might be relevant to avoid cases where the user stays indefinitely in any state.

Property:

$$“P_{max} = ?\left[F\left((Y_{11} = 1 \ \& Y_{12} = 1 \ \& \ Y_{13} = 1 \ )\& \ (q_1 = 1)\middle|(Y_{21} = 1 \ \& Y_{22} = 1 \ \& \ Y_{23} = 1 \ )\& \ (q_2 = 1) \ \middle|(Y_{31} = 1 \ \& Y_{32} = 1 \ \& \ Y_{33} = 1 \ )\& \ (q_3 = 1)\right)\right]”$$

Description: Max probability that in future any of User 1,2 or 3 will visit all the places and getting Quarantined.

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0	0.3282	0.5520	0.6895	0.7587	0.7778
	0.2	0	0.2921	0.4943	0.6209	0.6864	0.7050
	0.4	0	0.2412	0.4115	0.5207	0.5790	0.5962
	0.6	0	0.1689	0.2912	0.3721	0.4169	0.4306
	0.8	0	0.0724	0.1267	0.1641	0.1856	0.1926
	1	0	0	0	0	0	0

Table 6.7: Data of Fairness Property1

Graphical Representation:

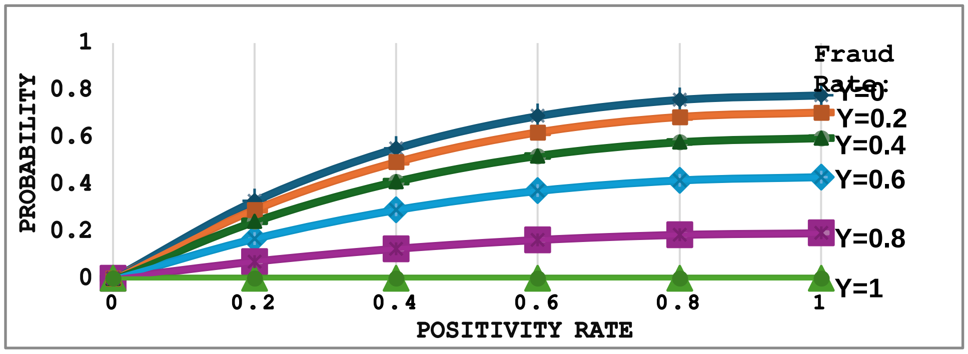


Figure 6.5: Graphical Representation of Fairness Property 1

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 0.8) as X increases from 0.2 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate.On the other hand, the probability values show decreasing trend for Y=0 to Y=0.8 for values of X (0.2 to 1) . In this case, the chances of getting quarantined even after visiting all places decreases with the increase in fraud rate.

6.4.2 Property 2:

“$P_{max} = ?\,[\,F\left(\left(Y_{11} = 1\ \&\ Y_{21} = 1\ \&\ Y_{31} = 1\right)\&\left(\left(z_1 = 0\ \&\ q_1 = 1\right)\middle|\left(z_2 = 0\ \&\ q_2 = 1\right)\middle|\left(z_3 = 0\ \&\ q_3 = 1\right)\right)\right]$”

Description: Max probability that in future any of the user are using same local transport and any of them is getting Quarantined without being Covid-19 positive.

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0	0.198847	0.322027	0.387623	0.413719	0.418402
	0.2	0	0.163292	0.265312	0.320413	0.342947	0.347272
	0.4	0	0.120221	0.196378	0.238437	0.256363	0.260126
	0.6	0	0.070022	0.115459	0.141493	0.153312	0.1561
	0.8	0	0.020553	0.034511	0.043038	0.047301	0.048464
	1	0	0	0	0	0	0

Table 6.8: Data of Fairness Property 2

Graphical Representation:

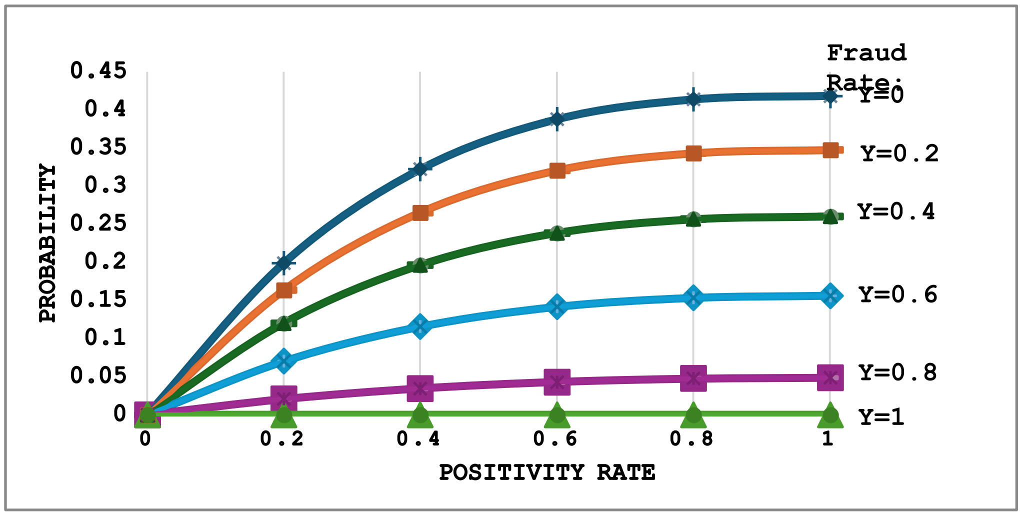


Figure 6.6: Graphical Representation of Fairness Property 2

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 0.8) as X increases from 0.2 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0 to Y=0.8 for values of X (0.2 to 1). According to the above results, the chances for a user with COVID-19 negative result will increase with the decrease in fraud rate, while the user had previously visited any place.

6.5 Concurrency

In the context of the multi-user system, concurrency refers to the ability of the model to handle the execution of multiple user processes (modules) potentially in parallel.

Concurrency allows the model to simulate situations where users can perform actions (e.g., visiting restaurants, using local transport) concurrently or interleaved with each other. Individual users might be in different states (e.g., shopping, quarantined) at the same time, and their transitions between states can potentially happen in parallel. Concurrency becomes crucial when users compete for shared resources (e.g., limited space in restaurants). The model should handle scenarios where multiple users might try to access the same resource concurrently. Concurrency allows the model to capture the real-world behavior of multiple users interacting in a dynamic environment. The model can be extended to handle more users without significant changes, as concurrency enables parallel execution of user processes. It can be used for the model to analyze the impact of concurrency on system performance (e.g., waiting times, resource utilization).

6.5.1 Concurrent Quarantine Completion

This property is relevant to concurrency because it considers multiple users potentially completing quarantine concurrently. The property checks for scenarios where any of the three users (s_1, s_2, s_3) could be in quarantine and eventually leave concurrently (through the three branches). The OR operator suggests that the quarantine completion for each user can happen independently through potentially different paths.

Property:

$$\text{“}P_{max} = ?\,[\,F\,((s_1 = P_{lt}\ \&\ q_1 = 1)\ \&\ (E\,[F\,(q_1 = 0)])\ |\ (\,s_2 = P_{lt}\ \&\ q_2 = 1)\ \&\ (E\,[F\,(q_2 = 0)])\ )|(\,s_3 = P_{lt}\ \&\ q_3 = 1)\ \&\ (E\,[F\,(q_3 = 0)])\,]\text{”}$$

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0	0.3282	0.5520	0.6895	0.7587	0.7778
	0.2	0	0.2921	0.4943	0.6209	0.6864	0.7050
	0.4	0	0.2412	0.4115	0.5207	0.5790	0.5962
	0.6	0	0.1689	0.2912	0.3721	0.4169	0.4306
	0.8	0	0.0724	0.1267	0.1641	0.1856	0.1926
	1	0	0	0	0	0	0

Table 6.9: Data of Concurrency Property 1

Graphical Representation:

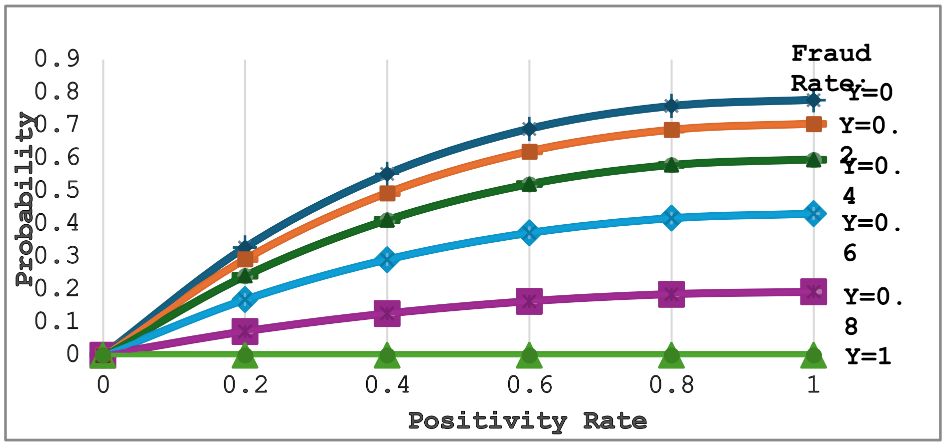


Figure 6.7: Graphical Representation of Fairness Property 1

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 0.8) as X increases from 0.2 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0 to Y=0.8 for values of X (0.2 to 1). The results shows that the user will eventually get out of quarantine with high chances, if the fraud rate is kept low.

6.5.2 Users Concurrently Visiting Restaurant

This property focuses on a specific scenario involving three users entering a restaurant together (s_1=P_{res} & s_2=P_{res} & s_3=P_{res}) and at least one user finishing their visit immediately in the next state (X (s1=P_{res1} | s2=P_{res1} | s3=P_{res1})).

It only considers a single situation where all three users enter a restaurant together. It doesn't explore other concurrent activities users might be engaged in or how they interact with resources concurrently. The property emphasizes the timing of state transitions, specifically that at least one user finishes their visit immediately after entering (X). This doesn't necessarily reflect overall concurrency across the system.

Property:

$$“P_{max} = ?\,[\,F\,((s_1 = P_{res}\,\&\,s_2 = P_{res}\,\&\,s_3 = P_{res})\,\&\,(\,X\,(s_1 = P_{res1}\,|\,s_2 = P_{res1}\,|\,s_3 = P_{res1})))\,]”$$

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0.1250	0.2162	0.3092	0.3793	0.4124	0.4184
	0.2	0.1111	0.1821	0.2609	0.3254	0.3582	0.3647
	0.4	0.0937	0.1424	0.2028	0.2583	0.2899	0.2970
	0.6	0.0714	0.0967	0.1334	0.1741	0.2015	0.2090
	0.8	0.0417	0.0476	0.0584	0.0750	0.0910	0.0970
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.10: Data of Concurrency Property 2

Graphical Representation:

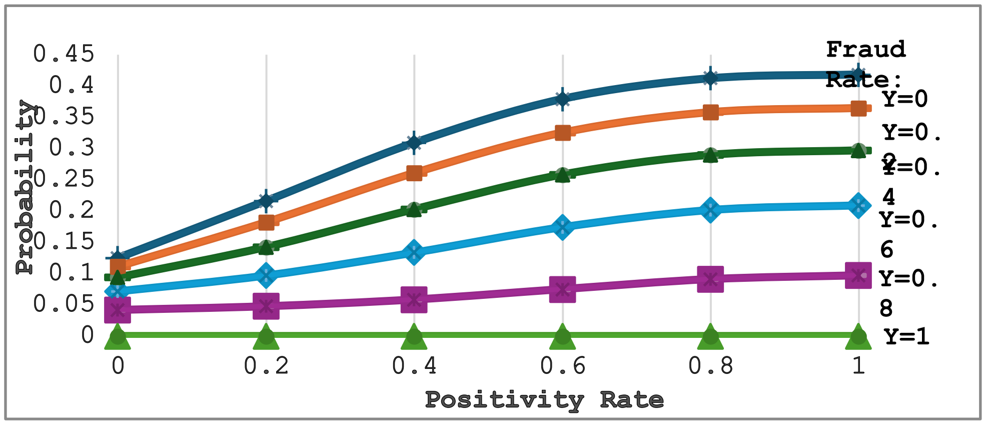


Figure 6.8: Graphical Representation of Concurrency Property 2

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 0.8) as X increases from 0 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0 to Y=0.8 for values of X (0 to 1). It is very clear from the results that the decrease in fraud rate will help users to follow valid path and avoid the invalid paths.

6.6 Safety of System

In n the context of the multi-user system you've described, safety refers to ensuring the model operates within desired boundaries and avoids harmful or unintended behavior. These are constraints that the system must always satisfy, regardless of the specific execution path taken. They often focus on preventing undesirable states or sequences of events. In PRISM, they are typically expressed using temporal logic operators like F (eventually), G (always), and negation (!).

The model should prevent scenarios where infection spreads rapidly and uncontrollably through the user population. These are situations where users get stuck waiting for each other or constantly engage in unproductive interactions, hindering progress. If the model includes limited resources (e.g., hospital beds), safety properties might ensure they don't become completely depleted. (e.g., waiting times, resource utilization).

6.6.1 System Ensuring Safety of System

This property uses F (eventually), which means it checks for the maximum probability of a path where User1 eventually becomes infected (z_1=1). The property doesn't guarantee that infection is prevented. It only looks for the maximum probability of a path leading to infection.

- It calculates the maximum probability (P_{max}=?) of a path where User1 eventually gets infected (z_1=1).
- The contact flags (Y_{11}, Y_{12}, Y_{13}) set to 1 indicate potential exposure to infected users 1, 2, or 3.

Property:

$$“P_{max} = ?\,[\,F((\,Y_{11} = 1 \mid Y_{12} = 1 \mid Y_{13} = 1 \;\&\; z_1 = 1))\,”$$

Empirical Data:

		X (Positivity Rate)					
		0	**0.2**	**0.4**	**0.6**	**0.8**	**1**
Y (Fraud Rate)	**0**	0.0000	0.3053	0.5142	0.6454	0.7178	0.7500
	0.2	0.0000	0.2752	0.4683	0.5944	0.6685	0.7059
	0.4	0.0000	0.2349	0.4058	0.5235	0.5990	0.6429
	0.6	0.0000	0.1792	0.3174	0.4206	0.4946	0.5455
	0.8	0.0000	0.1023	0.1893	0.2628	0.3242	0.3750
	1	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000

Table 6.11: Data of Safety Property 1

Graphical Representation:

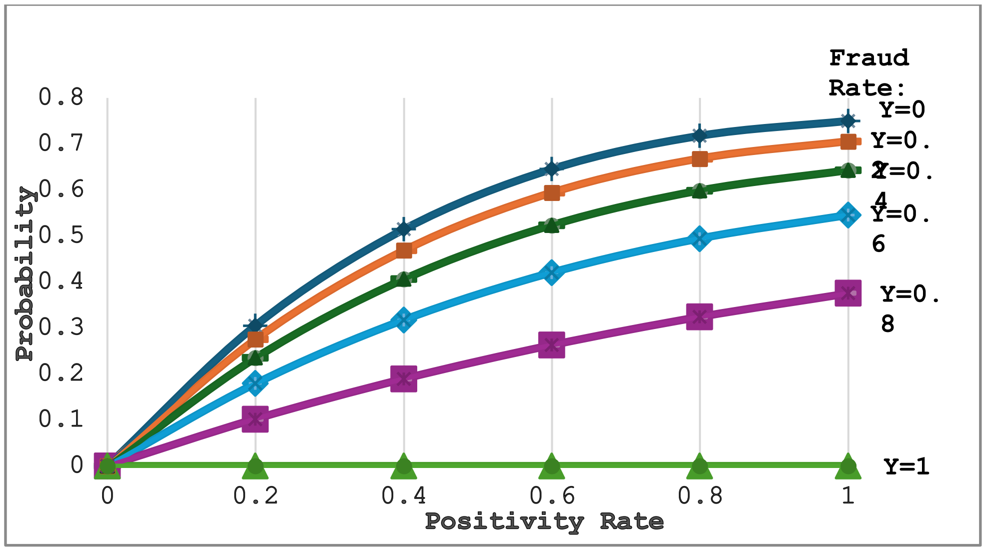


Figure 6.9: Graphical Representation of Safety Property 1

Discussion on result:

The probability values show an increasing trend for each values of Y (0 to 0.8) as X increases from 0.2 to 1.0. This means that the probability of a specific outcome increases as the positivity rate of a user increases, regardless of the fraud rate. The probability values show decreasing trend for Y=0 to Y=0.8 for values of X (0.2 to 1). The results shows that the user will eventually get out of quarantine with high chances, if the fraud rate is kept low. The results shows the safety of system is all related to fraud rate, the user visitig a place will be accurately quarantined, if all the users follow the valid path and get infected at any stage.

CHAPTER 7: CONCLUSION AND FUTURE WORK

In this chapter, we will sum up our discussion and give an overall idea of our system and conclude it. Moreover, we have presented the future work which is mainly based on the limitations of our work.

7.1 Conclusion

Our thesis presents the use formal verification for the Chinese health code system, which proved to be an effective tool for controlling the spread of infectious diseases, including COVID-19. The system's success can be attributed to the use of technology, including health QR codes and facial recognition, to monitor the movements of individuals and ensure compliance with quarantine measures. However, the system also has some loopholes, and it is important to address these concerns when implementing similar systems in other countries. Formal analysis of the Chinese health code system using PRISM provides us some valuable insights into its effectiveness and help identify limitations and areas for improvement.

Areas for Improvement:

- **Enhanced Validation:** Implement rigorous validation checks for user input and chosen paths. Ensure these checks align with the intended system design and functionality.

- **Review State Transitions:** Carefully review the logic governing transitions between states. Eliminate any unintended paths or inconsistencies that users could exploit.

- **Enhanced Enforcement:** Location data can be used to verify user compliance with designated paths or restrictions within the system.

- **Improved Monitoring:** Knowing user locations allows for better monitoring of user interactions and potential risks.

- **Data-driven Insights:** Location data can be analyzed to understand user movement patterns and inform system optimization.

We have explored various aspects of the system, focusing on concurrency, safety, reachability, reliability and fairness and potential areas for improvement.

While verifying different properties in the context of the above-mentioned points, we have some concluding remarks for them all. These findings suggest the importance of considering both user progress and the logic behind quarantine protocols within the model.

In the context of system Fairness, we verified property that focuses on preventing unrealistic user behavior by avoiding infinite loops within states. Secondly, identifying a potential issue with quarantine logic related to shared transport. As, this system is for multiple users so, we worked on their concurrent behavior and focused on a specific context (quarantine completion) and shows the system's ability to handle it. Secondly, highlights a limitation in the analysis scope for broader concurrency aspects.

This thesis highlights the importance of considering both the probability of infection and the potential sources of exposure when designing and evaluating the system. It suggests that further investigation might be needed to explore how to minimize the probability of such paths or identify additional factors influencing users' infection risk.

Key areas for improvement include enhanced validation, reviewing state transitions, refining the state representation, and formal verification. Adding location tracking offers benefits like enhanced enforcement and improved monitoring, but it also introduces challenges like privacy concerns and technical complexities. One can consider factors like technology choice, data management, rule definition, and data analysis when implementing location tracking.

7.2 Verifying a sample property for Some Insights:

In this part we will discuss the setup for experimentation and the limitations we are facing at this stage. As, the model insights can be gained by verifying a property for different combinations of positivity rate for COVID-19 and fraud rate at a particular area. Table 7.1

shows the statistics of model checking, here each row shows stats for different number of users. We have verified the same property, while keeping different number of users, sample property is given below. Here, the growth of states is increasing enormously with the increment in users count, and it is very clear that the increase in the number users will demand high computer resources. This is making it difficult for us to verify properties of this model for more than 3 users at this stage of development.

Sample Property:

"$P_{max} = ?\,[\,F((\,Y_{11} = 1 \mid Y_{12} = 1 \mid Y_{13} = 1\,)\,\&\,(\,q_1 = 1))$"

RAM Consumption:

Memory consumption is growing exponentially with the increase in count of users. It can be seen in Figure 7.1

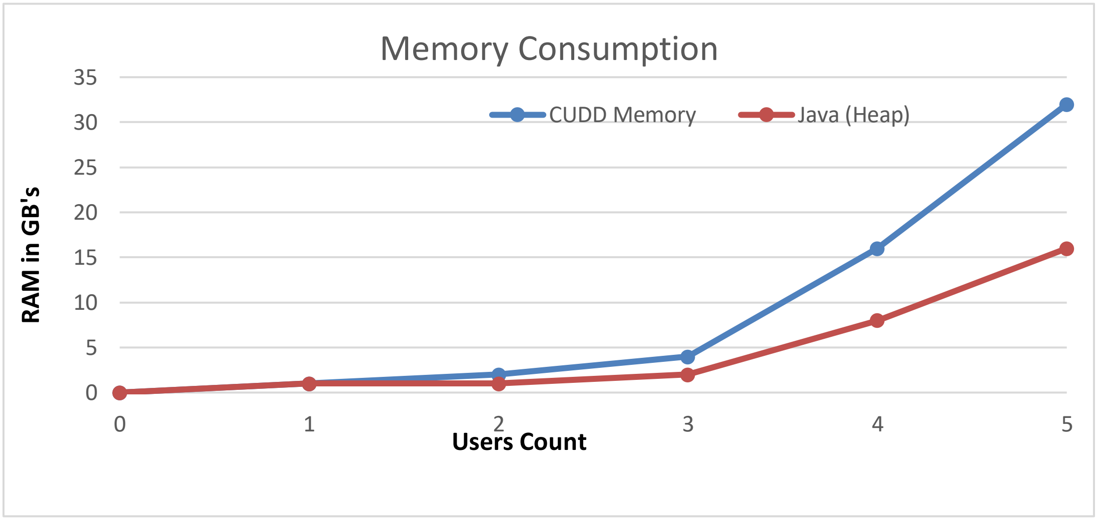


Figure 7.1: RAM consumed against the count of users

Time for model construction and model Checking:

Time required for model checking varies with different properties. It can be seen in Figure 7.2, that it is growing exponentially. Here, in our case it has taken 2 days (172,800 Seconds) and the process was killed by OS.

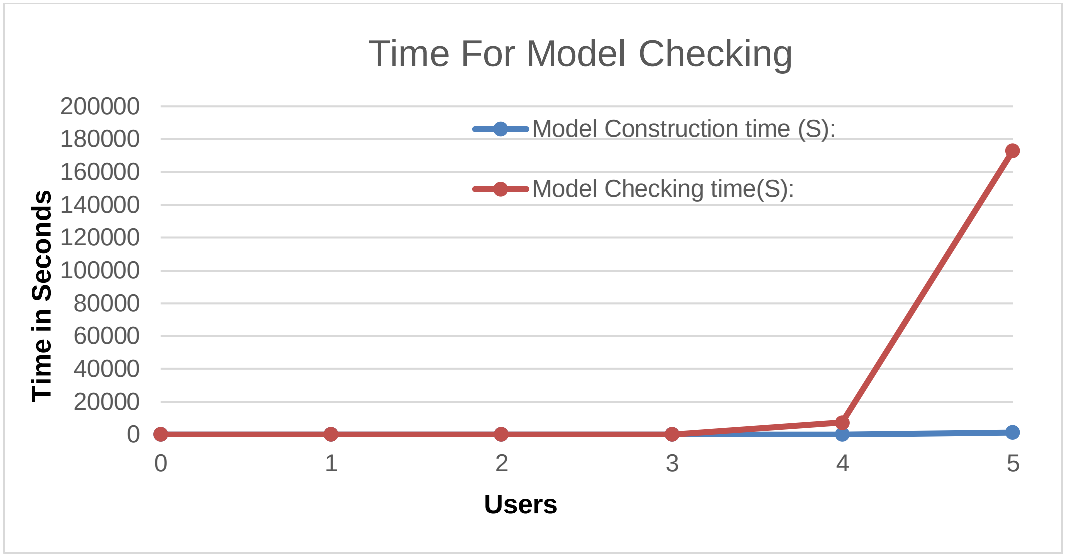


Figure 7.2: Increase in time consumption with increase in Users

Growth of States/ Transitions:

The growth of Sates and Transitions is also exponential, as shown in Figure 7.3 and Figure 7.4. This growth results in a state-based explosion while verifying property for 5 users.

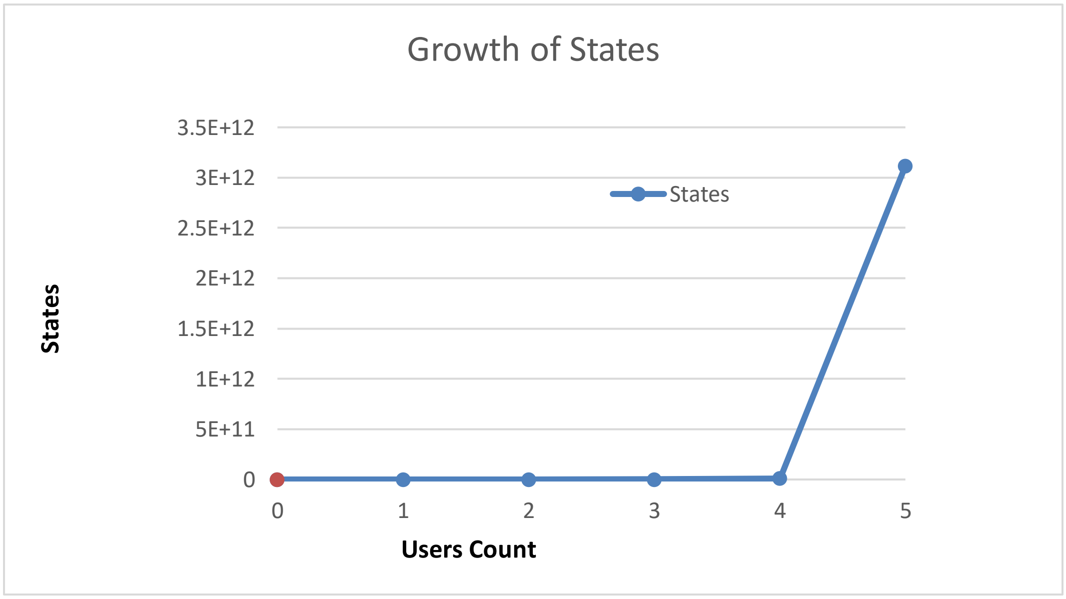


Figure 7.3: Growth of States with increase in User count

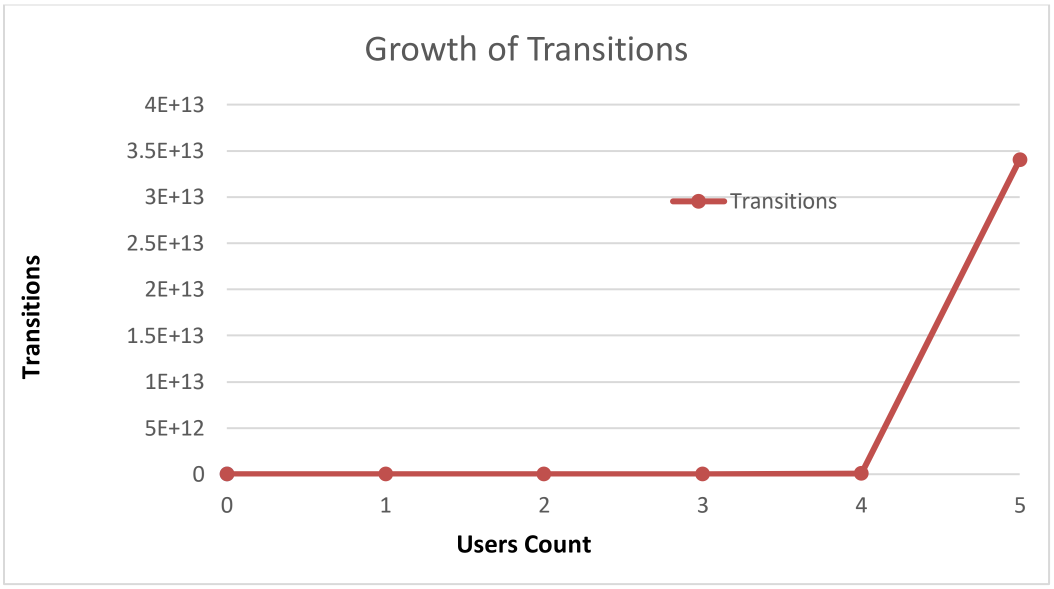


Table 7.4: Growth of Transitions with increase in User count

7.3 Future Work

Further research is needed to evaluate the long-term impact of the Chinese health code system and its potential as a model for other countries to adopt in their efforts to control the spread of infectious diseases. Overall, the system demonstrates the potential of technology in public health surveillance, and its success highlights the importance of balancing public health objectives with individual privacy concerns.

Here are some specific areas for future work that build on the initial foundation laid in this research:

7.3.1 Converting the model to real-time (PTA)

Currently, the model is not designed for real-time operation. Future work should explore how the model can be converted to a real-time probabilistic timing analysis (PTA) framework. This would allow for more immediate verification and improve the system's responsiveness to changes in the environment. The PTA framework could incorporate probability distributions of factors like disease transmission rates and testing turnaround times.

7.3.2 Covering more aspects of the model through diverse properties

The current model focuses on a limited set of properties. Future work should investigate how to incorporate additional properties into the model to provide a more comprehensive verification process. This could involve considering factors such as device security, data privacy, and user compliance with quarantine measures. Additionally, the model could be expanded to analyse the probability distributions of various outcomes based on these properties. For example, the model could assess the likelihood of a data breach under different security protocols.

7.3.3 Enhancement of the model for more than 3 users

The current model is designed for three users. Future work should investigate how the model can be adapted to handle verification for more than three users, such as entire more than a single community. This would require addressing scalability challenges and ensuring the efficiency of the verification process, while also incorporating probabilistic distributions of user behaviour. For instance, the model could account for the likelihood of different user compliance levels with health protocols ring the absence.

By addressing these areas of future work, we can continue to improve the effectiveness and robustness of formal verification techniques for public health applications, while also gaining a deeper understanding of the probabilistic nature of such systems.

REFERENCES

[1] H. Bohnenkamp, P. R. D'Argenio, H. Hermanns, and J.-P. Katoen, "MODEST: A Compositional Modeling Formalism for Hard and Softly Timed Systems," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 812–830, Oct. 2006, doi: 10.1109/TSE.2006.104.

[2] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston, "Automatic verification of real-time systems with discrete probability distributions," *Theor Comput Sci*, vol. 282, no. 1, pp. 101–150, Jun. 2002, doi: 10.1016/S0304-3975(01)00046-9.

[3] C. Eisentraut, H. Hermanns, and L. Zhang, "On Probabilistic Automata in Continuous Time," in *2010 25th Annual IEEE Symposium on Logic in Computer Science*, IEEE, Jul. 2010, pp. 342–351. doi: 10.1109/LICS.2010.41.

[4] F. Liang, "COVID-19 and Health Code: How Digital Platforms Tackle the Pandemic in China," *Soc Media Soc*, vol. 6, no. 3, p. 205630512094765, Jul. 2020, doi: 10.1177/2056305120947657.

[5] W. Chen, G. Huang, and A. Hu, "Red, yellow, green or golden: the post-pandemic future of China's health code apps," *Inf Commun Soc*, vol. 25, no. 5, pp. 618–633, Apr. 2022, doi: 10.1080/1369118X.2022.2027502.

[6] X. Zhang, "Decoding China's COVID-19 Health Code Apps: The Legal Challenges," *Healthcare*, vol. 10, no. 8, p. 1479, Aug. 2022, doi: 10.3390/healthcare10081479.

[7] G. Huang, A. Hu, and W. Chen, “Privacy at risk? Understanding the perceived privacy protection of health code apps in China,” *Big Data Soc*, vol. 9, no. 2, p. 205395172211351, Jul. 2022, doi: 10.1177/20539517221135132.

[8] W. Cong, “From Pandemic Control to Data-Driven Governance: The Case of China’s Health Code,” *Front Polit Sci*, vol. 3, Apr. 2021, doi: 10.3389/fpos.2021.627959.

[9] J. Shang, S. Wei, J. Jin, and P. Zhang, “Mental Health Apps in China: Analysis and Quality Assessment,” *JMIR Mhealth Uhealth*, vol. 7, no. 11, p. e13236, Nov. 2019, doi: 10.2196/13236.

[10] Y. Zou and J. Di, “Health Code as ‘access infrastructure’: Innovative practices and concerns of mediated governance,” *Global Media and China*, vol. 8, no. 3, pp. 381–413, Sep. 2023, doi: 10.1177/20594364231184110.

[11] J. Jiang and Z. Zheng, “Personal Information Protection and Privacy Policy Compliance of Health Code Apps in China: Scale Development and Content Analysis,” *JMIR Mhealth Uhealth*, vol. 11, pp. e48714–e48714, Nov. 2023, doi: 10.2196/48714.

[12] N. C. Büyükkaramikli, M. P. M. H. Rutten-van Mölken, J. L. Severens, and M. Al, “TECH-VER: A Verification Checklist to Reduce Errors in Models and Improve Their Credibility,” *Pharmacoeconomics*, vol. 37, no. 11, pp. 1391–1408, Nov. 2019, doi: 10.1007/s40273-019-00844-y.

[13] N. Gasteiger *et al.*, "Conducting a systematic review and evaluation of commercially available mobile applications (apps) on a health-related topic: the TECH approach and a step-by-step methodological guide," *BMJ Open*, vol. 13, no. 6, p. e073283, Jun. 2023, doi: 10.1136/bmjopen-2023-073283.

[14] S. Lagan, L. Sandler, and J. Torous, "Evaluating evaluation frameworks: a scoping review of frameworks for assessing health apps," *BMJ Open*, vol. 11, no. 3, p. e047001, Mar. 2021, doi: 10.1136/bmjopen-2020-047001.

[15] W. van der Toorn *et al.*, "An intra-host SARS-CoV-2 dynamics model to assess testing and quarantine strategies for incoming travelers, contact management, and de-isolation," *Patterns*, vol. 2, no. 6, p. 100262, Jun. 2021, doi: 10.1016/j.patter.2021.100262.

[16] A. Aleta *et al.*, "Modelling the impact of testing, contact tracing and household quarantine on second waves of COVID-19," *Nat Hum Behav*, vol. 4, no. 9, pp. 964–971, Aug. 2020, doi: 10.1038/s41562-020-0931-9.

[17] M. Salath *et al.*, "COVID-19 epidemic in Switzerland: on the importance of testing, contact tracing and isolation," *Swiss Med Wkly*, Mar. 2020, doi: 10.4414/smw.2020.20225.

[18] F. Yang, L. Heemsbergen, and R. Fordyce, "Comparative analysis of China's Health Code, Australia's COVIDSafe and New Zealand's COVID Tracer Surveillance Apps: a new corona of public health governmentality?," *Media International*

Australia, vol. 178, no. 1, pp. 182–197, Feb. 2021, doi: 10.1177/1329878X20968277.

[19] J. Martins, R. Barbosa, N. Lourenço, J. Robin, and H. Madeira, “Online Verification through Model Checking of Medical Critical Intelligent Systems.”

[20] Institute of Electrical and Electronics Engineers., *2014 IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom).*

[21] “tl-18-13”.

[22] M. Kwiatkowska, G. Norman, and D. Parker, “Probabilistic Model Checking and Autonomy,” 2022. [Online]. Available: www.annualreviews.org

[23] M. Kwiatkowska, G. Norman, and D. Parker, “Probabilistic model checking in practice: Case studies with PRISM 1.”

[24] M. Thesis and I. S. Zapreev David N Jansen Mariëlle IA Stoelinga, “Probabilistic model checking A comparison of tools,” 2007.

[25] G. Norman and D. Parker, “Quantitative Verification Formal Guarantees for Timeliness, Reliability and Performance A Knowledge Transfer Report from the London Mathematical Society and Smith Institute for Industrial Mathematics and System Engineering QUANTITATIVE VERIFICATION Formal Guarantees for Timeliness, Reliability and Performance,” 2014.

[26] M. Kwiatkowska, G. Norman, and D. Parker, “LNCS 6806 - PRISM 4.0: Verification of Probabilistic Real-Time Systems.”

[27] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang, "INFAMY: An Infinite-State Markov Model Checker," Springer-Verlag, 2009. [Online]. Available: http://depend.cs.uni-sb.de/

[28] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang, "A Model Checker for Parametric Markov Models," Springer-Verlag, 2010.

[29] E. M. Hahn, H. Hermanns, B. Wachter, and L. Zhang, "LNCS 6015 - <TEX>{\sf PASS}</TEX>: Abstraction Refinement for Infinite Probabilistic Models," Springer-Verlag, 2010.

[30] B. Jeannet, P. R. D'argenio, and K. G. Larsen, "Rapture: A tool for verifying Markov Decision Processes." [Online]. Available: http://www.irisa.fr/prive/bjeannet/prob/prob.html.

[31] J. Berendsen, D. N. Jansen, and F. Vaandrager, "Fortuna: Model checking priced probabilistic timed automata," in *Proceedings - 7th International Conference on the Quantitative Evaluation of Systems, QEST 2010*, 2010, pp. 273–281. doi: 10.1109/QEST.2010.41.

[32] D. A. Parker, "IMPLEMENTATION OF SYMBOLIC MODEL CHECKING FOR PROBABILISTIC SYSTEMS," 2002.

[33] I. T. Bhatti and O. Hasan, "Formal Verification of a Fully Automated Out-of-Plane Cell Injection System," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, IEEE, Mar. 2020, pp. 111–116. doi: 10.1109/ISQED48828.2020.9137036.