Epidemic Analysis Using Traditional Model Checking and Stochastic Simulation

Kanishka Chauhan

A Thesis Submitted to

Indian Institute of Technology Hyderabad

In Partial Fulfillment of the Requirements for

The Degree of Master of Technology

Department of Computer Science Engineering

June 2015

Approval Sheet

This Thesis entitled Epidemic Analysis Using Traditional Model Checking and Stochastic Simulation by Kanishka Chauhan is approved for the degree of Master of Technology from IIT Hyderabad

[illegible]

U.RAMAKRISHNA Examiner
Dept. of ......
IITH

[illegible]

Dr. N.R.Aravind Examiner
Dept. of ...CSE
IITH

[illegible]

(Dr MVP Rao) Adviser
Dept. of CSE
IITH

[illegible]

Manish Singh Chairman
Dept. of ...CSE
IITH

Declaration

I declare that this written submission represents my ideas in my own words, and where ideas or words of others have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be a cause for disciplinary action by the Institute and can also evoke penal action from the sources that have thus not been properly cited, or from whom proper permission has not been taken when needed.

Kanishka

(Signature)

Kanishka Chauhan

(Kanishka Chauhan)

cs13m 1003

(cs13m1003)

Acknowledgements

At the outset, I would like to express my sincere gratitude to my thesis adviser **Dr MVP Rao** for his constant encouragement, and faith in my work. Without his guidance and persistent help, this dissertation would not have been possible.
Further, I would like to thank Mr M Moiez Gohar for the numerous fruitful conversations on TA, PTA. TCTL and Formal Methods concepts. I am also thankful to my friends who took pain of proofreading this dissertation. Finally, I thank almighty and my family for their support and constant encouragement.

Abstract

Stochastic model checking has been the mainstay for formal analysis of epidemic progression in recent years. However, such methods are sensitive to inaccuracies in estimating stochastic parameters like infection transmission and recovery rates. In this work, we revert to traditional model checking (specifically, for timed automata) to absorb inaccurately provided parameters into the non-determinism inherent in such traditional formalisms. Parameters obtained through stochastic simulation are used by the timed automata, with sufficiently wide windows of non-determinism to account for error. A positive side effect of this approach is that separating the probabilistic component from actual epidemic timed automata model, helps us to focus on the progression logic while building the model.

We demonstrate this approach using the model checker UPPAAL in conjunction with the stochastic epidemic modeling and simulation tool GLEAM.

Contents

Chapter 1

Introduction

1.1 Background

Mathematical modeling of epidemics helps in predicting the progression of an epidemic among populations and across geographies. This in turn, helps in planning measures to control it. Epidemic modeling has been studied extensively in the past. While the initial models were deterministic in nature [1], recognition of the fact that epidemic propagation is inherently stochastic shifted the focus to probabilistic models. Indeed, several epidemics have been modeled in this fashion in the past. Initial stochastic models assumed homogeneous mix in the populations each individual in a population is connected to every other individual in the same manner [2]. Therefore, an infected individual can transmit the virus to any other individual with the same probability. However, it was realized that this is not a very realistic model as individuals have limited contacts. Edge weighted contact networks have since been used to capture neighborhood [3]. Modeling and simulation efforts in these areas have yielded several interesting results as well as tools for studying the progression of epidemics. While simulations are useful for studying simple properties of the progression of an epidemic, more complex questions are difficult to answer. Moreover, a quantitative assurance of accuracy in the results is desirable.
With these in mind, researchers turned to statistical model checking. Mathematically precise formalisms (like Continuous Timed Markov Chains (CTMC)) are used to model epidemic progression among populations and queries regarding the model are framed in a suitable logic (like Probabilistic Computational Tree Logic (PCTL), Probabilistic Timed Computation Tree Logic (PTCTL), Continuous Stochastic Logic (CSL) etc). Following are some example queries of potential interest:

- What is the likelihood that at time t, at least c% of the population of a geographic region is affected?
- What is the likelihood that at time t, the carrier who imports the disease for the first time to a given geographic region arrives?

On the other hand, traditional model checking is done as follows. Given the description of a system in a mathematically precise formalism and the requirements expected of it as a formula in an appropriate (non probabilistic) system of logic, model checking algorithms attempt to verify if the described system satisfies the specified properties.

Among the two, statistical model checking gained traction in the field of epidemic analysis for several reasons. To begin with, the problem is fundamentally stochastic in nature and thus amenable to stochastic model checking. Secondly, this approach generally offers decent speed, if only at the expense of accuracy. And last but not the least, statistical model checking offers scope for more fine grained analysis. A statement like there is a 10% chance of the epidemic reaching a particular geographic region in the next 50 days is more useful than a simple affirmation that it is possible for the epidemic to reach in the next 50 days.

1.2 Motivation

We report an approach that combines traditional model checking with stochastic simulation methods. The motivations are following.

Firstly, this approach separates the stochastic element from the logic of epidemic progression and simplifies the process of designing the model. We appeal to stochastic simulations only to get an estimate of some of the delays required in the traditional model.

Secondly, it is often difficult to accurately establish various probabilistic parameters of an epidemic model. We compensate for the loss of accuracy by absorbing it into the *non determinism* available in the modeling formalism. Suppose that through stochastic simulation, we have estimates of the time t when $r\%$ of the population of a given geographic region gets infected by the epidemic. Then, our approach centers on allowing for these events to occur at any time during a window of time using non determinism for example, between $t + \delta t$ and $t - \delta t$. The size of the window of non determinism can be chosen based on ones confidence on the accuracy of the stochastic simulation. Such a use of non determinism has been reported in the past in different contexts [4].

A final motivation is that there are differences in the expressiveness of various logic systems necessitating the framing of some queries in non probabilistic logic. Having traditional model checking approaches as well in the repertoire plugs that gap.

Timed automata, defined by Alur and Dill [5], have proved to be very useful for modeling and verification of timed systems. Together with formal specifications required of the system (in a temporal logic) and algorithms for model checking, it provides a technique for analyzing temporal behaviour of such systems. Model checking tools like UPPAAL [6, 7] have been developed and widely used for the purpose [8, 9, 10, 11].

Our approach involves the temporal parameters like:

(i) Given that a certain geographic region is affected by the epidemic, how long does it take for the infection to appear in another region?
(ii) How long does it take for the infection to assume epidemic proportions in a region, and

(iii) How long does it take for a region to recover from the epidemic.

These parameters in turn depend on various socio-economic parameters like the connectivity between two regions, the strength of the healthcare system of the involved countries etc. For instance, the more connected a given country is to an affected country, the faster the epidemic reaches there. On the other hand, a strong healthcare system minimizes the duration for which a country remains affected. However, it is difficult to translate these parameters into the temporal parameters that we are looking for, for use in the timed automata model.

We get around this by leveraging the "Global Epidemic and Mobility model", a tool that integrates real world data of the kind described above with stochastic models of epidemic propagation.

Following are some example queries as per our approach. Given that an infection starts in a certain part of the world on day 0,

- Will a certain geographic region be necessarily infected between days 120 and 150? Choice of venues for mass international events like sports meets in a certain interval of time, and planning medical logistics.
- Is it guaranteed that there will be some date after day 200 when all geographic regions will be free of the infection?

1.3 Organization of Thesis

Thesis is arranged as follows. The next section provides a brief discussion of existing work in the area. Section 3.1 provides a brief introduction to Timed Automata, Tree Computation Tree Logic(TCTL), UPPAAL and the GLEAMviz tool. Chapter 4 describes our approach and discusses example queries and their results. Finally Chapter 5 concludes the work with discussion on challenges faced during modeling epidemic propagation.

Chapter 2

Related Work

In this section, we briefly review some important recent work in the area of epidemic model checking. As discussed in the previous section 1.1, most of the literature concerns with stochastic model checking. Drabik and Scatena [12] model the progression of an epidemic in a homogeneous population using the PRISM model checker and ask queries framed in PCTL. Sam Huang [13] strikes a compromise between homogeneous population models and contact networks among individuals by treating population that exhibit similar behavior as a node and establishing contact network between these nodes. This model is encoded as a CTMC and the queries are framed in CSL. However, it is not made clear how one obtains some crucial parameters, like the weights on the edges between two meta-nodes. While this is not crucial for demonstrating the efficacy of vaccination strategies (as was indeed the aim of this work), it is not clear how it models the progression of an epidemic in the real world. Both approaches use the PRISM model checker for analysis and report experiments for small scale scenarios.

Ciochetta and Hillston [14] report an approach on the Bio-PEPA process algebra. In this approach also, the population is divided into homogeneous sub populations and the progression of the epidemic across various contact network topologies is investigated. For model checking, they derive a PRISM model from the Bio-PEPA model and frame queries in PCTL. They report that for the derived PRISM model, the verification of properties is not very efficient. In all the above mentioned works, the reported experiments are small scale small populations divided among a small number of patches.

In a seminal work, Bortolussi and Hillston [15] use fluid approximation techniques in reducing the state space of the agent based CTMS for approximate model checking of epidemic propagation. The approach is complex, as the limit model of an individual is a time inhomogeneous CTMC.

Complex networks and social networks provide a close imitation of interactions of individuals. Therefore, as research in these areas matured, results therein have been increasingly applied in understanding epidemic dynamics, cf. [16, 17, 18, 19, 20] and references therein. This also led to advances in related problems like propagation of computer viruses across networks [21].

Chapter 3

Preliminaries

3.1 Timed Automata

Following is a brief discussion about Timed Automata (TA). Please read [5] for a detailed treatment. A timed automaton is essentially a finite state machine extended with clock variables. It uses a dense time model where a clock variable evaluates to a real number and all clock variables progress synchronously.

Definition 1: A Timed Automaton A is a tuple (Q, Σ, C, E, q_0) where

- Q, a finite set of states of A,
- Σ, a finite set called the alphabet or actions of A,
- C, a finite set called the clocks of A,
- $E \subseteq Q \times \Sigma \times B(C) \times P(C) \times Q$, a set of edges, called transitions of A, where,
  - $B(C)$ is the set of boolean clock constraints involving clocks from C, and
  - $P(C)$ is the powerset of C
- a function $\mathit{Inv} : I(C)$ that associates with each location a set of boolean constraints on the clocks.
- q_0, an element of Q, called the initial state.

Informally, the automaton can remain in a state only as long as it does not violate an *invariant* condition associated with that state. On the other hand, it can jump to another state across an edge if a *guard* condition associated with the edge is satisfied. These conditions are encoded in $B(C)$. On taking a jump, it is possible to reset a subset of clocks. Thus, an edge (l, σ, b, p, l') represents a transition from location l to l' , where σ is an action symbol and b is the guard condition associated with the edge, and p is the subset of clocks that need to be reset.

Such automata can be constructed to model real time reactive systems. An extremely useful fact is that it is possible to define parallel composition of timed automata. This facilitates writing timed automata models for complex systems as similar to modular approach in Software Industry.

Having constructed the timed automata model, one asks queries framed in a temporal logic regarding the temporal behaviour of such an automata. Model checking algorithms are designed to answer such queries. Some properties of interest are:

1. Reachability: If a specific condition holds in some state of the model.
2. Safety: If a specific condition holds in all states of an execution path.
3. Liveness: If a specific condition is guaranteed to hold eventually.
4. Deadlock: If there is a deadlock in the model.

Not surprisingly, safety and liveness are the most important properties. Fortunately, these have been shown to be decidable in case of timed automata [5].

There exist a variety of tools for the analysis of timed automata and its extensions, including the model checkers like UPPAAL [6, 7]. We are using UPPAAL for verification of our Timed automata model. UPPAAL is useful for verifying desirable properties of systems that can be modeled as networks of timed automata. Additionally, it implements integer variables, structured data types, user defined functions, and channel synchronisation and urgency in states. The query language of UPPAAL, that specifies the properties to be verified in the timed automata model, is a subset of Timed Computation Tree Logic (TCTL).

This subset includes queries of the form:

- $A <> \phi$: Always, ϕ will eventually hold true
- $A\,[\,]\,\phi$: Invariantly, ϕ is true
- $E <> \phi$: Eventually, ϕ will be true on some path
- $E\,[\,]\,\phi$: Potentially, ϕ is always true

where $\phi := A.L \ | \ g_d \ | \ g_c \ | \ \phi$ and $\phi \ | \ \phi$ or $\phi \ | \ $ not $\phi \ | \ \phi \Rightarrow \phi \ | \ (\phi)$ for automaton A. Location L, data guard g_d and clock guard g_c. Clock guards are predicates involving clocks: $x * c$ where $* \ \epsilon \ \{<, <=, ==, >=, >\}$ for clock variable x and natural number c. Data guards are predicates over standard arithmetic expressions:$E * E$ where E is a standard arithmetic expression $* \ \epsilon \ \{<, <=, ==, >=, >, != \}$. Finally, we also make use of the "bounded liveness" query provided in UPPAAL: $\phi -- > (\varphi$ and $x < c)$ which says that whenever ϕ is true, φ is true within c time units.

For the purpose of understanding Timed Automata modeling basics, Lamp-User [7] example is demonstrated below. Example problem specification is like this:
There is a "User" and "Lamp". "User" presses lamp button and "Lamp" has 3 locations viz. $Off, Low, Bright$.
Given that Lamp is in Off state,

Case-1: Single Click , on pressing the button once, Lamp should go to *Low* state.

Case-2: Double Click , on pressing the button twice quickly, Lamp should go to *Bright* state. Here quickly means, User has to press twice within 3 time units.

Case-3: Double Click , on pressing the button twice lazily, Lamp should go to Off state.

So, two major entities are involved namely "User" and "Lamp". "User" presses the lamp-button. "Lamp" detects that button is being pressed by "User". Depending on the applicable Case#, "Lamp" changes its state. Two timed automaton are designed separately namely Lamp automaton and User automaton, to capture the behavior of Lamp and User respectively. Whenever User presses the button, a signal is sent to Lamp automaton (using *press*!) and Lamp take transition to next state after listening to press signal (using *press*?). Here *press* is synchronisation channel/signal, where

- **press!** denotes
  press signal is sent to Lamp automaton, whenever this transition (Self loop labeled with press! in User Automaton, Figure 3.1) is taken by User automaton.
- **press?** denotes
  transition (transition/edge over which press? is labeled) is taken only on the receiving of press signal from sender automaton in system.

and these two transitions (one which is sending signal and one which is receiving) occur simultaneously at the same time. Clock variable c takes care of the timing restriction. So Whenever User presses the button, clock c gets reset and start ticking. If user presses quickly i.e. within less than 3 time units, *guard* condition ($c < 3$) allows Lamp automaton to go into *Bright* state. If user presses lazily, i.e. User is taking more than 3 time units. In that case *guard* condition ($c >= 3$) allows Lamp automaton to go into Off state. If Lamp is already in *Bright* state and User presses the button once, on receiving press signal, Lamp automaton goes into Off state. Figure 3.1 and 3.2 shows Lamp-User system modeled as two separate automaton in UPPAAL tool. UPPAAL tool itself does parallel composition of two automaton before doing model checking.

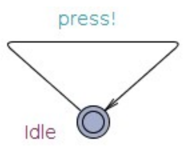

Figure 3.1: User Automaton

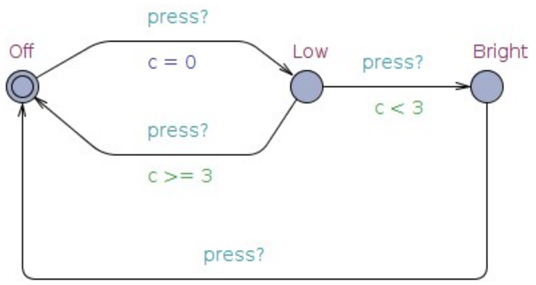

Figure 3.2: Lamp Automaton

Model Checking

1. Is there any deadlock in system?
   TCTL query: A [ ] not deadlock
   Verdict: *Satisfied.*

2. Is it the case that Lamp is in state *Bright* and clock value still higher than 3?
   TCTL query: E <> lamp.*Bright* and $c \geq 3$
   Verdict: *Not Satisfied.*

Above mentioned queries verdicts ensure that system is deadlock free and it is never the case that system is not following requirement specifications.

3.2 TCTL Model Checking

It is useful to know how model checking is performed by UPPAAL using TCTL queries. UPPAAL takes the model (*say* M) as input, which user designs as network of timed automaton. UPPAAL takes TCTL query (which is actually property to be verified on model). UPPAAL converts TCTL query into automaton (*say* P).
Implementation (Model M) meets specification P iff $L(M) \subseteq \; P$. Then Verification problem simply reduced to checking emptiness of $L(A_M \bigcap L(A_P)^c$.

3.3 The Infection Model

In epidemiology, infections are often characterized using the *compartmental model*. At any given point in time, each individual of the population is in one of several possible "compartments" - susceptible, latent, infected and symptomatic, infected but not symptomatic, recovered etc. The individual then transits from one compartment to another with time.

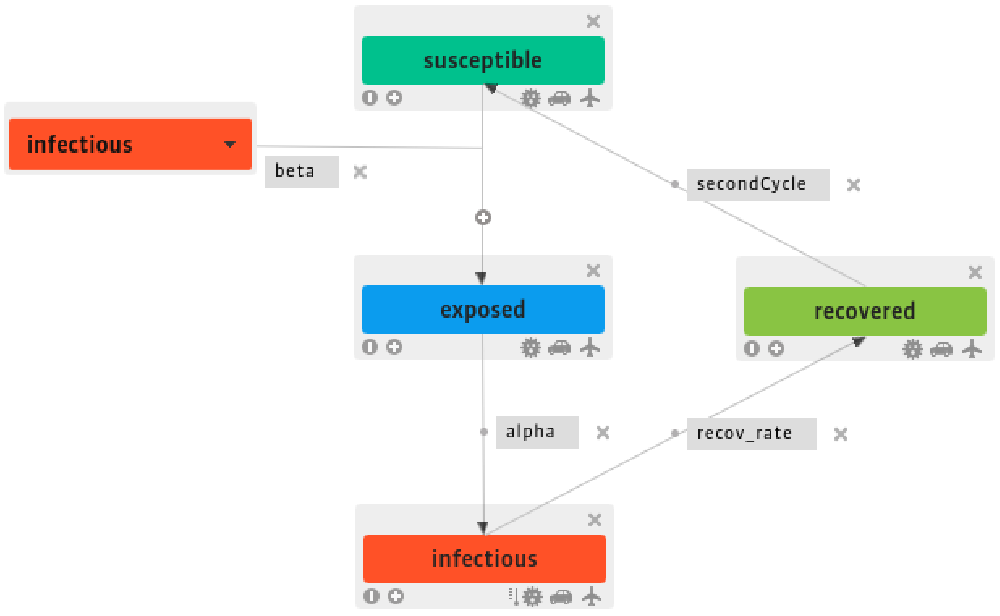

Figure 3.3: Compartmental Model of Infection: Edges labeled with transition rates

Together with interconnections between compartments, and the rates of inter-compartment transitions, one can characterize the infection. For example, an S-E-I-R-S model of infection makes an individual cycle through being susceptible, exposed, infected, recovered and being susceptible again.

To obtain values for the parameters to be used in the timed automata, we rely on the GLobal Epidemic and Mobility Model (available publically as the GLEAMviz tool [22]), which models the progression of an epidemic as a stochastic process. GLEAM takes as input this characterization of the infection, together with other parameters like places where the infection originates, the fraction of the population infected initially in these places, connectivity parameters like flight occupancy rates, commute times etc. GLEAM uses real-world data for populations, human mobility, and health care conditions in a locality, to enhance accuracy in prediction and analysis of the future course of an epidemic outbreak. The interested reader is referred to [23] for details.

Indeed, GLEAM has been used to predict the progression of epidemic outbreaks like the 2002-3 SARS [24], the 2009 H1N1 [25] and more recently, the Ebola outbreak of 2014 [26].

3.4 About GLEAMviz

The GLEAMviz client is a desktop application by which users interact with the GLEAMviz tool. It provides GUI for its four main functions:

1. the design of compartmental models that define the infection dynamics;
2. the configuration of the simulation parameters;
3. the visualization of the simulation results; and
4. the management of the users collection of simulations. i.e. user can see simulation results offline also.

3.4.1 Disease Model built using GLEAMviz

The Model Builder provides a graphical tool for designing compartmental model corresponding to infection. Model Builder enables the user to define the compartment label (susceptible/latent/infectious. etc), the mobility constraints applicable to each compartment (e.g. whether road, air commuting is allowed/not allowed) and initial fraction of population in each compartment.

The disease model is depicted in Figure 3.3. We can see that in disease model, road and air commuting is allowed in every compartment. There are only 4 compartments namely susceptible, exposed, infectious and recovered. Box labeled "infectious" between susceptible and exposed compartments actually means that people in susceptible compartment come in contact with infectious people with rate *beta* and become exposed. People in exposed compartment become infectious with expected rate *alpha*. Likewise people in infectious compartment get recovered with rate *recover_rate* and again become susceptible after certain period of time with rate *second_cycle*. This inter-compartmental movement of people is observed for 365 days (default setting in GLEAMviz tool), thus epidemic propagation behavior can be observed for 365 day duration.

3.4.2 GLEAMviz Visualization interface

Compartment model corresponding to infection is submitted to GLEAM server. Simulation runs on server and output is sent back to client. Geographic map is displayed as a result of simulation. Map can be zoomed to see city/country locations. On clicking a particular city, it's details get popped up, namely population of city, number of beds/hospital per 10000 people, age distribution of population. It allows to visualize propagation of epidemic from one place to another as people start commuting from one geographic region to another. In between epidemic statistics at particular date can be monitored like number of new cases reported on that day, total number of people infected till now, The visualization interface is depicted in Figure 3.4.

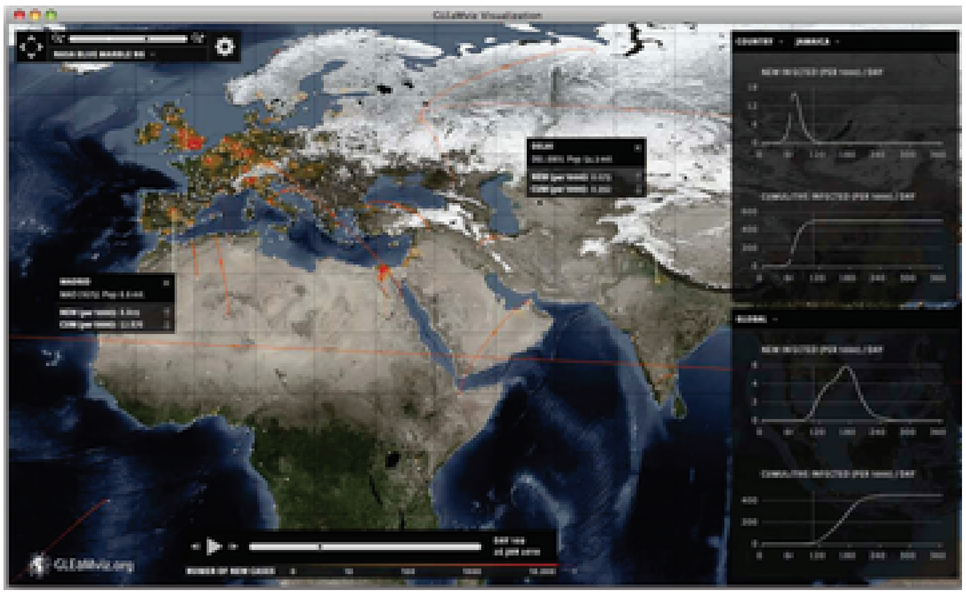

Figure 3.4: GLEAMViz: Simulation Visualization Interface

Chapter 4

Our Approach

4.1 Modeling epidemic dynamics

We model geographical regions as vertices of a complete graph - that is, there exists an edge between every pair of regions. This essentially means that every region is connected to every other region through different means of transport.

One of these regions is designated as the originator of the epidemic. Corresponding to each geographic region, we construct a timed automaton that models the temporal behaviour of the epidemic in that region. We use a slightly different automaton for the geographic region where the infection originates. The automata are shown in Figures 4.1 and 4.2.

Table 4.1 provides a look-up for the variables used in the automata.

Term	**Meaning**
id	region identifier
t[id]	clock associated with region $c[id]$.
e:id_t	non deterministic selection of region id
c[id].d[e]	delay at region $c[id]$ before the first occurrence of infection is brought in from $c[e]$
infected[i]	a boolean array that records whether a $region\ \ i$ is affected by the epidemic or not
infectCount	number of times region $c[id]$ goes into the epidemic phase in a given period of time
c[id].rt	time needed at $c[id]$ to recover from the epidemic
c[id].st	time after the first infection appears but before the region reaches its epidemic phase
Init(id)	resets a region's local parameters like $t[id]$ and $c[id].infected$
SeenB[ ]	an array for recording the regions from which epidemic alerts have appeared during the current epidemic cycle
SeenItBefore[id][e]	decides if an epidemic alert from region e has already been considered in the current cycle (makes use of seenB[])
gt	clock that keeps global time

Table 4.1: Notation

There are locations in each automaton that signify

- (i) when the region is **unaffected** by the epidemic (labeled by **A**, and a special addition one time starting location $Safe$ for the originating region)

- (ii) the time elapse at the region before the first infection appears because of some other region getting into its epidemic phase (labeled by B, D, F, H, J; detailed explanation later in the section)
- (iii) the time elapse between the first appearance of the infection (carried in from abroad) in the region and its reaching the **epidemic stage** (labeled by **K**) and
- (iv) the time taken for the region to **recover from the epidemic** (labeled by **L**).

We first describe a simplistic scenario for ease of exposition. The automaton for every region is initially in its unaffected location A. The automaton of a region, say R_1, leaves this location only if some other region (say R_2) gets into the epidemic phase. The movement out of the unaffected location is triggered by "alerts" from R_2 in the form of synchronization and shared variables provided in UPPAAL. Even after R_2 gets into the epidemic phase, it takes some time for the infection to reach R_1 through a carrier. This delay, say d_{21}, depends primarily on the connectivity from region R_2 to R_1[1]. Moreover, let us assume an inaccuracy of δ in the stochastic estimation of d_{21} through GLEAMviz stochastic tool. Then, the automaton for R_1 waits nondeterministically in location B for a delay between $d_{21} - \delta$ and d_{21} time units. After this delay, the first incidence of the infection in R_1 occurs due to its connectivity with R_2 . It takes st time units (Spread Time) for R_1 to enter into epidemic phase. Assuming the error of δ' time units, the automaton waits nondeterministically in location K for a delay between $st - \delta'$ and st units. On completion of this delay, it transits to location L. This location signifies the full blown epidemic. On non deterministically completing between rt (Time to Recover) and (a conservative) $rt + \delta''$ time units in the "epidemic" L phase, the automaton returns to the unaffected location A if all other regions are uninfected. Otherwise, the infection cycle starts all over again for the region. As before, δ'' is the error that we anticipate in the stochastic simulation.

The delays st and rt for a region to recover depends on the strength of its healthcare system. Parameters like the number of hospital beds per 10000 people are good indicators of this strength.

This simplistic scenario can get complicated when another region (say R_3) with which R_1 has better connectivity gets into its epidemic phase while R_1 is still waiting for the infection to be brought in from R_2 . Let the delay of getting the infection from R_2 be d_{31}. Further, let t_1 be the time at which region R_1 becomes susceptible on account of R_2 getting into its epidemic phase. Let the first infection carrier come to R_1 from R_2 at $t_1 + d_{21}$. Suppose at t such that $t_1 \leq t \leq t_1 + d_{21}$, R_3 enters into its epidemic phase. If $t_1 + d_{21} - t \geq d_{31}$, a infection carrier from R_3 will arrive at R_1 earlier than one from R_2 , and the automaton changes location to reflect this. Otherwise, the automaton maintains status quo and waits for d_{21} units for the carrier from R_2. Note that multiple regions can enter into their epidemic locations at different points in time, potentially modifying the time of first arrival of a infection carrier. Locations B, D, F, H and J capture thisone, two, three, four and five regions getting into their respective epidemic location. These parameters d_{ij}, st and rt have been determined using data collected from GLEAMviz [22], an epidemic modeling tool in section 4.2.

[1]While the connectivity in terms of air, road or sea transport from R_2 to R_1 is expected to be the same in the opposite direction, we do not assume this.

4.2 Determining parameters using GLEAMviz

It remains to determine values of various parameters in the timed automata. As mentioned earlier, we use the GLEAM model and simulator for this purpose. We choose an SEIRS infection model with parameters as shown in Table 4.3. The simulation is run for a period of 365 days.

To obtain the delays d_{ij} , we choose i as the originating region and measure the delay before an infection appears first in region j. Doing this for all pairs in both the directions, we obtain the "Delay" columns of Table 4.2. The "spread time" is defined as the number of days elapsed between the first appearance of the infection in a region and 1% of the population getting infected. This is when the region enters the epidemic phase. Similarly, the "recovery time" is the number of days elapsed between entering the epidemic phase and the first day when no new infection appears [2]. These times are recorded; see columns labeled "Spread time" and "Recovery time" in Table 4.2.

Continent	**Delay d_{ij}**						**Spread time**	**Recovery time**
Asia	2	18	4	23	8	8	40	167
Africa	10	2	6	22	18	20	65	175
Europe	6	6	2	6	4	23	57	170
South America	27	21	10	2	8	24	76	169
North America	8	18	4	2	2	8	60	174
Australia	8	20	21	32	15	2	61	172

Table 4.2: Parameters of the system

In this discussion, we choose geographic regions at the granularity of continents for brevity of explanation. Thus, the world is divided into the six major continents. GLEAMviz allows a choice of finer granularity as well, namely subcontinental regions etc. For each continent, we identify some important, well connected cities as the origin of the infections. These values serve as parameters to the timed automata implemented in UPPAAL. In the timed automata, we choose Asia as the region where the infection originates in the first place.

4.3 Queries and Results

The queries posed and the results obtained upon checking them against the model in Uppaal are depicted in Table 4.4. Table 4.4 shows some example queries and their answers. Queries 1 and 2 concern all the regions in different intervals of time, while 3 and 4 concern a specific region. For example, query number 1 asks whether at any time between day 300 and day 400 all the regions are necessarily affected by the epidemic. Similarly, query number 3 asks whether region 3 is necessarily affected between day 200 and day 300. For these queries, we have taken $\delta = 4$ and $\delta' = \delta'' = 1$. A change in these values would change verdicts accordingly.

[2]These definitions can be changed as required.

Continent	**Originating cities**
Asia	Beijing, Shanghai, Tokyo, Seoul, Mumbai
Africa	Lagos, Cairo, Johannesberg, Nairobi
Europe	London, Paris, Madrid, Rome, Berlin, Istanbul
South America	Sao Paulo, Rio de Janeiro, Buenos Aires, Lima
North America	New York, Los Angeles, Chicago, Mexico City, Toronto
Australia	Sydney, Melbourne, Brisbane
Comapartment transition rates (SEIRS)	
Susceptible-Latent	0.835, 0.417 Due to Infectious and Asymtomatic Infectious respectively
LatentInfectious and Symptomatic	2/3.3
LatentInfectious but Asymptomatic	1/3.3
Infected and SymptomaticRecovered	0.6
Infected but AsymptomaticRecovered	0.6
RecoveredSymptomatic	0.0001
Miscellaneous	
Start Date	18/12/2014
Flight Occupancy Rate	90%
Time Spent at commuting destination	8 hrs
Fraction of population infected at origin (Mumbai)	0.00042

Table 4.3: GLEAM Parameters

S. No.	**TCTL Query**	**Verdict**
1	**A**<> (gt >= 300 and gt <= 400 and infected[0] == 1 and infected[1] == 1 and infected[2] == 1 and infected[3] == 1 and infected[4] == 1 and infected[5] == 1)	N
2	**A**<> (gt >= 200 and infected[0] == 0 and infected[1] == 0 and infected[2] == 0 and infected[3] == 0 and infected[4] == 0 and infected[5] == 0)	N
3	**A**<> (gt >=200 and gt <= 300 and infected[3] == 1)	Y
4	**A**<> (gt >= 120 and gt <= 150 and infected[3] == 0)	Y

Table 4.4: Example Queries

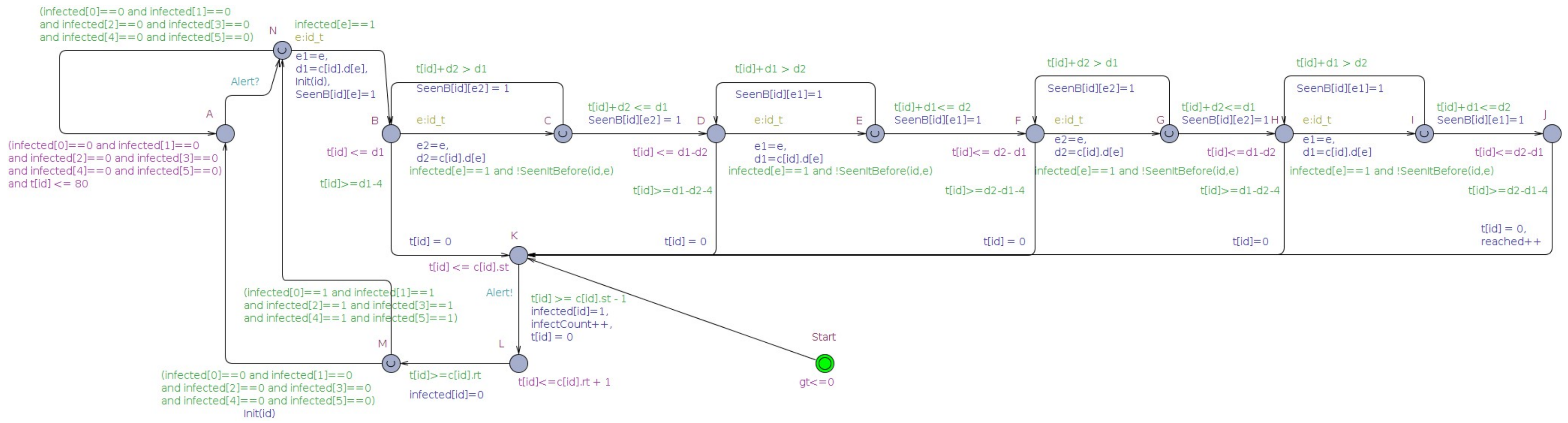

Figure 4.1: Timed Automata for first Infected region

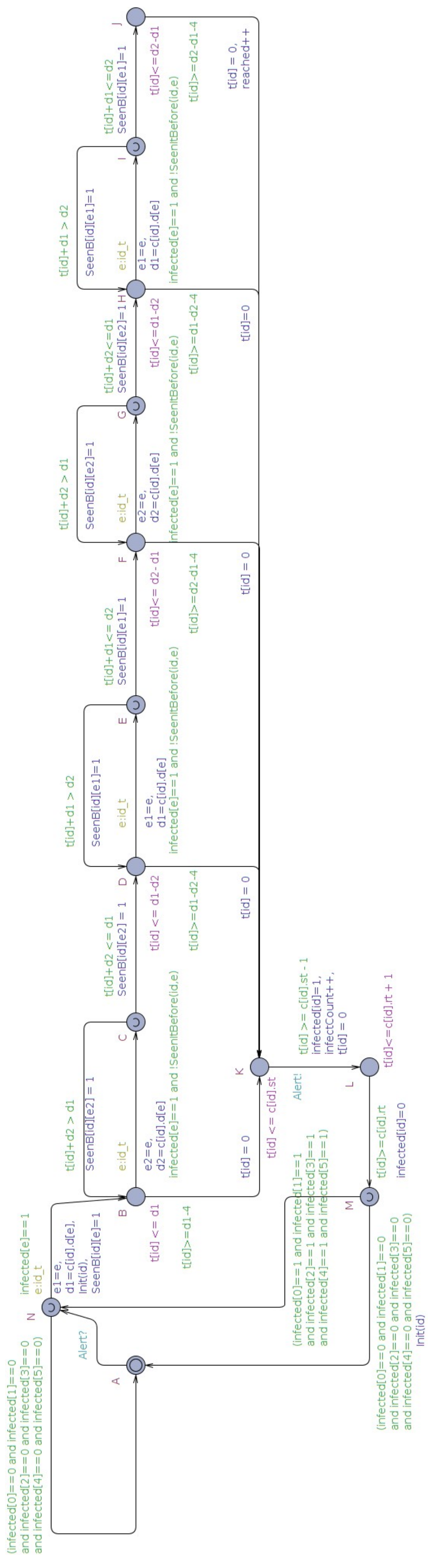

Figure 4.2: Timed Automata for all other regions

Chapter 5

Conclusion

Current model is designed with 6 continents into consideration. This number can be scaled to large number of locations as per requirement of problem definition. But checking large location timed automata model may lead to state space explosion and verifying such model is really problematic as large memory and more time is needed to verify properties. Sometimes it is observed that some queries never terminate. This is one major problem in scaling the model to more number of geographic regions because TCTL model checking is exhaustive by nature.

One may overcome this state space explosion problem by cleverly reusing variables, number of clocks, synchronisation channels and every data structure construct that is mutable. Best is to reduce their number as much as possible. But doing so is not possible always.

In our epidemic model, we have kept number of geographic regions minimum due to this very reason. But at the same time, to maintain generality of model and practical enough to mimic real epidemic propagation , we choose 4 to 5 number of major cities in every continent.

The underlying concepts and modeling formulations of timed automata can be applied to other network models, viz. social networks, computer networks (to study spread of viruses), and so on and so forth.

Bibliography

[1] Hyman G Landau and Anatol Rapoport. Contribution to the mathematical theory of contagion and spread of information: I. spread through a thoroughly mixed population. *The bulletin of mathematical biophysics*, 15(2):173–183, 1953.

[2] Helen Abbey. An examination of the reed-frost theory of epidemics. *Human biology*, 24(3):201–233, 1952.

[3] Lauren Meyers. Contact network epidemiology: Bond percolation applied to infectious disease prediction and control. *Bulletin of the American Mathematical Society*, 44(1):63–86, 2007.

[4] Luca De Alfaro, Thomas A Henzinger, and Ranjit Jhala. Compositional methods for probabilistic systems. In *CONCUR 2001Concurrency Theory*, pages 351–365. Springer, 2001.

[5] Rajeev Alur and David L Dill. A theory of timed automata. *Theoretical computer science*, 126(2):183–235, 1994.

[6] Kim G Larsen, Paul Pettersson, and Wang Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.

[7] Gerd Behrmann, Alexandre David, and Kim G Larsen. A tutorial on uppaal. In *Formal methods for the design of real-time systems*, pages 200–236. Springer, 2004.

[8] Magnus Lindahl, Paul Pettersson, and Wang Yi. Formal design and analysis of a gear controller. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 281–297. Springer, 1998.

[9] Johan Bengtsson, Kim Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. *UPPAALa tool suite for automatic verification of real-time systems*. Springer, 1996.

[10] Johan Bengtsson, WO David Griffioen, Kåre J Kristoffersen, Kim G Larsen, Fredrik Larsson, Paul Pettersson, and Wang Yi. Verification of an audio protocol with bus collision using uppaal. In *Computer Aided Verification*, pages 244–256. Springer, 1996.

[11] Fredrik Larsson, Paul Pettersson, and Wang Yi. *On memory-block traversal problems in model-checking timed systems*. Springer, 2000.

[12] Peter Drábik and Guido Scatena. An application of model checking to epidemiology. *Proceedings of Applications of Membrane computing, Concurrency and Agent-based modelling in Population biology (AMCA-POP)*, pages 90–97, 2010.

[13] Samuel Huang. Probabilistic model checking of disease spread and prevention. *Scholarly Paper for the Degree of Masters in University of Maryland*, 2009.

[14] Federica Ciocchetta and Jane Hillston. Bio-pepa for epidemiological models. *Electronic Notes in Theoretical Computer Science*, 261:43–69, 2010.

[15] Luca Bortolussi and Jane Hillston. Fluid model checking. In *CONCUR 2012–Concurrency Theory*, pages 333–347. Springer, 2012.

[16] Matt J Keeling and Ken TD Eames. Networks and epidemic models. *Journal of the Royal Society Interface*, 2(4):295–307, 2005.

[17] Yang Wang, Deepayan Chakrabarti, Chenxi Wang, and Christos Faloutsos. Epidemic spreading in real networks: An eigenvalue viewpoint. In *Reliable Distributed Systems, 2003. Proceedings. 22nd International Symposium on*, pages 25–34. IEEE, 2003.

[18] Marcelo N Kuperman. Invited review: Epidemics on social networks. *Papers in Physics*, 5:050003, 2013.

[19] Mark EJ Newman. Spread of epidemic disease on networks. *Physical review E*, 66(1):016128, 2002.

[20] Maria Deijfen. Epidemics on social network graphs. *Unpublished master's thesis, Stockholm University, Sweden*, 2000.

[21] Deepayan Chakrabarti, Yang Wang, Chenxi Wang, Jurij Leskovec, and Christos Faloutsos. Epidemic thresholds in real networks. *ACM Transactions on Information and System Security (TISSEC)*, 10(4):1, 2008.

[22] info [at] gleamviz.org. Gleamviz.

[23] Wouter Broeck, Corrado Gioannini, Bruno Goncalves, Marco Quaggiotto, Vittoria Colizza, and Alessandro Vespignani. The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale. *BMC Infectious Diseases*, 11(1):37, 2011.

[24] V Colizza, A Barrat, M Barthelemy, and A Vespignani. Predictability and epidemic pathways in global outbreaks of infectious diseases: the sars case study. *BMC Medicine*, 5:34, 2007.

[25] Paolo Bajardi, Chiara Poletto, Jose J Ramasco, Michele Tizzoni, Vittoria Colizza, Alessandro Vespignani, et al. Human mobility networks, travel restrictions, and the global spread of 2009 h1n1 pandemic. *PloS one*, 6(1):e16591, 2011.

[26] Marcelo FC Gomes, Ana Pastore y Piontti, Luca Rossi, Dennis Chao, Ira Longini, M Elizabeth Halloran, and Alessandro Vespignani. Assessing the international spreading risk associated with the 2014 west african ebola outbreak. *PLoS currents*, 6, 2014.