Bayesian statistical model-checking of continuous stochastic logic

by

Pavithra Prabhakar

Ph.D., University of Illinois at Urbana-Champaign, 2011

A REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Statistics
College of Arts and Sciences

KANSAS STATE UNIVERSITY
Manhattan, Kansas

2022

Approved by:

Major Professor
Christopher Vahl

Copyright

Abstract

Autonomous systems are transforming the society by enabling sophisticated technologies such as robotic surgery and driverless cars. On one hand, increased automation through removal of the human-in-the-loop promises enhanced efficiency, while, on the other hand, the highly uncertain and safety critical environments, such as, varying weather and road conditions, and the presence of pedestrians on the road, pose challenge to the design of reliable autonomous systems. Hence, there is an immediate need for a robust framework for certifying the correctness of autonomous systems.

In this report, we explore verifying the correctness of uncertain autonomous systems modeled as discrete-time Markov chains (DTMCs) against correctness criteria provided as continuous stochastic logic (CSL) formulae. Statistical model-checking (SMC) is a paradigm for verification based on formulating the verification problem as a hypothesis testing problem. We propose a novel statistical model-checking algorithm based on Bayesian hypothesis testing. While Bayesian approaches for simpler logics without nested probabilistic operators and Frequentist approaches for nested logic have been previously explored, the Bayesian approach for CSL that has nested probabilistic operators has not been addressed. The challenge in the nested case arises from the fact that unlike in probabilistic model-checking (PMC), where we obtain a definitive answer for the model-checking problem for the sub-formulae, we only obtain a correct answer with a certain confidence, which needs to be factored into the recursive SMC algorithm. We have implemented our algorithm in a Python Toolbox, and present our evaluation on some benchmark examples. We observe that while both the Bayesian and frequentist SMC perform well in terms of inference, Bayesian SMC is more efficient in terms of the number of samples. On several examples, it even beats the state-of-the-art probabilistic model-checker PRISM.

Table of Contents

List of Figures

List of Tables

Acknowledgments

First, I would like to thank my major professor Christopher Vahl for providing me the opportunity to do the M.S. report under his guidance. I would like to thank professors Juan Du and Haiyan Wang for serving on my MS report committee, and teaching several courses that I thoroughly enjoyed.

This report is a collaborative work with Ratan Lal, a former PhD student in the computer science department and Weikang Duan, a PhD student in the statistics department at Kansas State University. I believe that this is a truly foundational interdisciplinary work that spans computer science and statistics and would not have been possible without the hard work, dedication and collaboration of Ratan and Weikang.

I would like to thank the professors from the Department of Statistics who taught the several courses that built up my foundations in statistics. I would like to thank the department of computer science, department head Scott DeLoach and Kansas State University for the support toward pursuing the MS in Statistics.

Finally, I would like to thank my family who have perpetually encouraged and supported me in all my endeavors. A special thanks to my husband Som for being my strength in all my pursuits.

Dedication

To Moshi
for being my strength

Chapter 1

Introduction

Autonomous systems are transforming the society by enabling sophisticated technologies such as robotic surgery and driverless cars. On one hand, increased automation through removal of the human-in-the-loop promises enhanced efficiency, while, on the other hand, the highly uncertain and safety critical environments, such as, varying weather and road conditions, and the presence of pedestrians on the road, pose challenge to the design of reliable autonomous systems. Hence, there is an immediate need for a robust framework for certifying the correctness of autonomous systems.

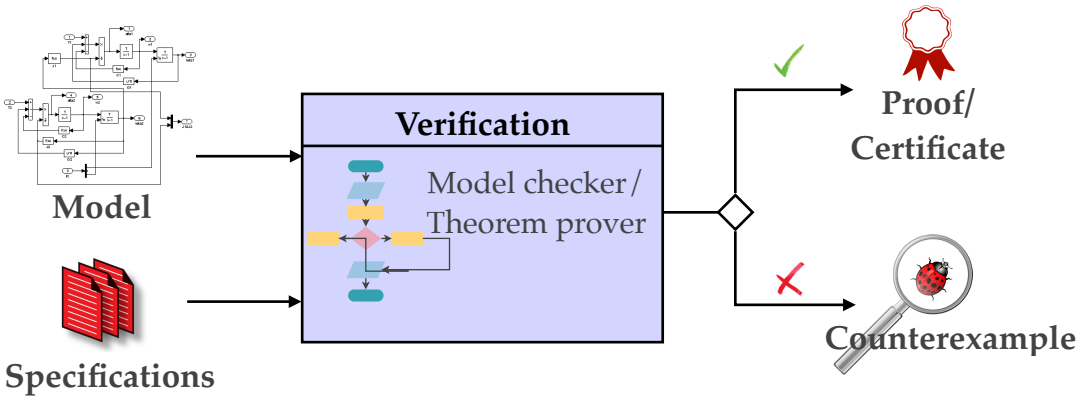

Figure 1.1: *Overview of Formal Verification*

Formal verification is an area of computer science that deals with scalable and rigorous analysis of systems by automatically constructing proofs of correctness. Formal verification

(see Figure 1.1) consists of (a) a mathematical model of the system to be analyzed, typically provided as an automaton or a transition system, (b) an unambiguous specification of correctness, typically provided as a logical formula and (c) a verification algorithm that takes the model and the specification as input and generate a proof that the model satisfies the specification. Verification algorithms can be broadly classified as those based on model-checking and theorem proving. Model-checking algorithms are based on state-space exploration and are typically fully automated, while theorem proving often requires manual interference to complete the proofs.

In this report, we consider the verification task, where the model is provided as a Discrete-time Markov Chain (DTMC) and the correctness specification is provided in a probabilistic logic called Continuous Stochastic Logic. Autonomous systems interact with uncertain environments, hence, stochastic models are required to capture their behavior. In this work, we focus an a simple stochastic transition system model, namely, DTMCs which consist of states and probabilistic transitions between states. For instance, in[1], the stochastic behavior of a car driver is modeled as a DTMC, and several properties about the driver behavior modeled in the logic Probabilistic Computation Tree Logic (PCTL) are verified. Probabilistic Model-Checking (PMC)[2] is a verification paradigm that provides an exact answer to the model-checking problem involving probabilistic models and specifications. More precisely, PMC returns an exact answer to the verification question: given a DTMC $\mathcal{T}$ and a formula φ, does $\mathcal{T}$ satisfy φ? For instance, the formula $P_{\geq 0.9}(F_{\leq k}\, \mathit{Target})$ states that the probability of reaching the state *Target* in the next k steps is at least 0.9. A probabilistic model checking algorithm will return true if the sum of probabilities of paths that reach the state *Target* within k steps is at least 0.9 in the given DTMC. Probabilistic model checking has been explored extensively for various probabilistic models such as DTMC, Markov Decision Processes, Continuous-time Markov Chains, Timed and Hybrid Probabilistic Systems as well as several probabilistic logics including Probabilistic Computation Tree Logic (PCTL) (See related works section).

While probabilistic model checking has made tremendous progress in verification of probabilistic systems, scalability is still an issue when the state-space becomes large. Statistical

model-checking[3–12] is an alternate technique that provides a scalable method that instead of state-space exploration, determines the satisfiability of a formula based on samples and statistical tests. However, these methods cannot provide exact answers, but only provide correct answers with certain confidence specified using Type I and Type II errors that specify the probability of an incorrect answer when the system satisfies/violates the property, respectively. For instance, consider the formula $P_{\geq\theta_0}(F_{\leq k}\,\mathit{Target})$. We can rewrite this as a hypothesis testing problem with hypotheses $H_0 : \theta \geq \theta_0$ vs $H_1 : \theta < \theta_0$ called the null and alternate hypothesis, respectively. Statistical model checking algorithm consists of simulating random paths of the DTMC, counting the number of paths that satisfy $\rho = (F_{\leq k}\,\mathit{Target})$ and using a statistic based on the counts to deduce whether to accept/reject H_0. It is possible that a hypothesis is rejected even when it is true, or vice versa. These errors are captured using Type I and Type II errors using values α and β, that provide upper bounds on the probability of rejecting H_0 when it is true, and accepting H_0, when it is false, respectively. Different statistical hypothesis testing algorithms provide different methods to control the α and β errors. In SMC, there is the frequentist approach in which the parameters are assumed to be fixed (unknown) and the Bayesian approach in which parameters are assumed to have a probability distribution (prior).

In this work, we present a Bayesian statistical model-checking (SMC) algorithm for verifying Continuous Stochastic Logic (CSL) specifications on discrete-time Markov Chain (DTMC) models. Bayesian SMC has been explored previously for a fragment of CSL without nested probabilistic operators[4;5]. A frequentist SMC approach based on acceptance sampling has been explored for the complete CSL logic[3]. Here, we build upon ideas from these papers to address the problem of Bayesian SMC for the complete CSL logic. Nested probabilistic operators are challenging in the case of SMC. Consider a nested formula $P_{\geq 0.9}(F_{\leq k}P_{\geq 0.5}G_{\leq n}\,\mathit{Target})$, which says that the probability of reaching a state from which the probability of being in the *Target* for n consecutive steps is 0.5, is 0.9. While probabilistic model-checking can verify such formulas inductively by first annotating each state with the satisfiability of $P_{\geq 0.5}G_{\leq n}\,\mathit{Target}$ and then using that information to verify the rest of the formula. The challenge of adopting a similar approach for SMC arises from the fact

that we cannot infer the definitive answers to the satisfaction of the formula $P_{\geq 0.5} G_{\leq n}\, Target$ at a particular state, but can only obtain the answer with certain confidence given by α and β, which need to be factored into the recursive algorithm. We tackle this by basing our test on a statistic based on these uncertain answers, quantifying its deviation from exact answers, and finally, approximating its values by computable lower and upper bounds to obtain an algorithm.

We have implemented a simplified version of our algorithm in a Python toolbox and compared with the frequentist SMC algorithm and probabilistic model checking (PMC) algorithm. In terms of computation time, our algorithm beats both the approaches. Further, our Bayesian SMC approach requires fewer number of samples than the frequentist SMC approach for deducing the correctness. All the approaches provide correct answers in our experiments. In addition, we compared our Bayesian SMC approach with respect to different priors. We observed that though uniform prior beats the other choices, the choice of prior did not lead to significant differences in the performance, which eliminates the disadvantage of the Bayesian approach which is criticized for the subjectiveness in the choice of the prior.

Chapter 2

State-of-the-art

There are broadly two approaches for verification of probabilistic models, namely, probabilistic model-checking and statistical model-checking. Probabilistic model-checking (PMC) deduces whether a probabilistic systems satisfies a property by an exhaustive state-space exploration, while statistical model-checking (SMC) uses a sampling based approach to deduce the correctness. PMC often provides an answer with full confidence, while SMC provide an answer with certain Type I and Type II errors. We provide a brief overview of the related works for both these approaches.

Probabilistic model-checking algorithms have been explored for different classes of models and specification logics. DTMCs are finite state systems whose transitions consist of a probability distribution on the next states, MDPs allow both non-deterministic and probabilistic transitions, and CTMCs consist of transitions with probabilistic timing information. Several approaches for verification of DTMCs have been explored including an automata-theoretic approach to verify LTL on probabilistic (concurrent) finite-state programs[13;14], model-checking algorithms for the probabilistic extension of the logic CTL (Computation Tree Logic)[15;16], namely, PCTL[17–19], and symbolic algorithms for DTMC analysis based on the symbolic representation of Algebraic Decision Diagrams and MTBDDs (multi-terminal BDDs)[20–23]. Quantitative reachability verificaiton for MDPs by successive refinement[24], value iteration[25], and linear programming[26;27] based approaches, as well as model-checking

MDPs with respect to probabilistic extensions of branching time logics such as pCTL* have been explored[28]. CTMC model-checking has been shown to be decidable with respect to Continuous Stochastic Logic (CSL) using results in algebraic and transcendental number theory[29]. The problem of model-checking CSL is shown to be reducible to solving a systems of linear equations, a Volterra integral equation system and to computing transient state probabilities of CTMCs[30]. An approximation algorithm for verifying branching-time properties of CTMC has been explored[30]. Abstraction based analysis techniques that construct simpler/smaller probabilistic systems have been explored for DTMCs, MDPs and CTMCs[31–34].

Several tools have been developed that implement probabilistic model-checking algorithms. PRISM[35] is a well-known probabilistic model-checker that implements model-checking algorithms for DTMC/MDPs with respect to PCTL specifications, CTMCs with respect to CSL specifications, and models with probabilistic, non-deterministic and real-time characteristics[2;36–40]. In addition, several other model-checkers have been developed for verification of probabilistic properties specified in probabilistic computation tree logic (PCTL) and continuous stochastic logic (CSL) and several probabilistic models, such as, ETMCC[41], MRMC[36], and YMER[42]. A comparative analysis of the performance of these model checkers based on verification time and peak memory usage has been performed[43]. Satisfiability Modulo Theory (SMT) based approach has been developed for analyzing infinite Markov decision processes[44].

An alternate approach to PMC is Statistical Model Checking (SMC) that infers the correctness of the probabilistic systems by evaluating certain test statistics on random samples (paths) generated from the probabilistic models. Different methods based on frequentist and Bayesian approach have been developed for SMC. A frequentist hypothesis testing based algorithm for SMC of CSL on Discrete-time Markov chain (DTMC) based on sequential Wald Probability Ratio Test[45] has been explored[3]. A Bayesian SMC algorithm to check a stochastic model with well-defined probability space over traces (e.g DTMC, CTMC) against properties specified in *Probabilistic Bounded Linear Temporal Logic* (PBLTL) has been explored[4]. In addition, SMC algorithms for black-box systems has been implemented

in the tool VESTA[8;46]. A Bayesian approach for verification of DTMCs against non-nested CSL specification has been explored[4;5], this report explores its extension to the nested case. Also, Bayesian estimation and Bayesian interval methods for Bayesian SMC for stochastic discrete-time hybrid system model against PBLTL specifications have been explored[5].

Furthermore, SMC for complex models such as Generalized Semi Markov Process (GSMP)[8], Generalized Stochastic PetriNet (GSPN)[47], stochastic hybrid systems[12], priced Timed Automata[10], and Markov Decision Processes[11] have been explored. Several surveys on Statistical Model-Checking provide an exhaustive list of related literature[6;7].

Chapter 3

Verification of Discrete-Time Markov chains

In this chapter, we formalize the verification problem we study in this report, wherein we model the uncertain systems as discrete-time Markov chains (DTMCs), and specify the correctness criteria using continuous stochastic logic (CSL) formulae.

3.1 Discrete Time Markov Chains

Discrete Time Markov Chains (DTMCs) are probabilistic models that consist of a finite set of states along with transition probabilities that specify the probability of a state change/transition. Let $\mathcal{D}ist(S)$ denote the probability distributions over a finite set S, that is, functions $p : S \to [0, 1]$ such that $\sum_{s \in S} p(s) = 1$.

Definition 1. *A Discrete Time Markov Chain (*DTMC*) is a structure* $\mathcal{T} = (\mathcal{S}, \longrightarrow, AP, L)$*, where:*

- $\mathcal{S}$ *is a finite set of states;*
- $\longrightarrow \subseteq \mathcal{S} \times \mathcal{D}ist(S)$ *is transition relation such that for any state* $s \in \mathcal{S}$*, and* $\rho_1,\ \rho_2 \in \mathcal{D}ist(\mathcal{S})$*, if* $(s, \rho_1),\ (s, \rho_2) \in \longrightarrow$*, then* $\rho_1 = \rho_2$*;*

- *AP is a set of atomic propositions; and*
- $L : \mathcal{S} \rightarrow 2^{AP}$ *is a labelling function.*

An edge $(s, \rho) \in \longrightarrow$ *will also be denoted as* $s \longrightarrow \rho$.

Next, a finite (infinite) path of DTMC $\mathcal{T} = (\mathcal{S}, \longrightarrow, AP, L)$ is a finite (infinite) sequence of states $\sigma = s_0 s_1 s_2 s_3 \ldots$ such that there exists a finite (infinite) sequence of probability distributions $\rho_0 \rho_1 \rho_2 \rho_3 \ldots$ such that $s_i \longrightarrow \rho_i$, and $\rho_i(s_{i+1}) \neq 0$, for all i. We use $\sigma[i]$ to denote the i-th state in the path, namely, s_i. In addition, for a finite path $\sigma = s_0 s_1 s_2 \ldots s_n$, $len(\sigma)$ denotes the length of the path σ, that is, $len(\sigma) = n$. Let $\mathcal{P}aths(\mathcal{T})$ denote the set of all possible infinite paths of $\mathcal{T}$.

We can define a σ-algebra on $\mathcal{P}aths(\mathcal{T})$ in a standard manner (see, for instance,[48]). The σ-algebra is defined by assigning probabilities with finite paths each of which specifies a cylinder consisting of all infinite paths that contain the finite path as a prefix. All measurable sets are built from the cylinders using the operations of complementation and countable union, and the probabilities are accordingly assigned. Given a finite path $\sigma = s_0 s_1 s_2 \ldots s_n$, $prob(\sigma)$ denoting the probability of path σ, is defined as follows.

$$prob(\sigma) = \prod_{i=0}^{n-1} \rho_i(s_{i+1}).$$

Given a measurable set of paths $\mathcal{C}$ of the DTMC $\mathcal{T}$, $prob(\mathcal{C})$ denotes the probability of $\mathcal{C}$ in the σ-algebra.

3.2 Continuous Stochastic Logic (CSL)

In this section, we discuss continuous stochastic logic, a logic for expressing probabilistic properties. It essentially extends the branching-time logic, namely, Computation Tree Logic (CTL*), by replacing the existential and universal quantifications over paths by probabilistic operators. Next, we present the syntax and semantics of CSL.

Syntax Given a set of atomic propositions AP, a CSL formula φ over AP, is inductively defined as:

$$\star\ \varphi ::= tt \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid Pr_{\bowtie\theta}(\varrho); \quad \star\ \varrho ::= X\varphi \mid \varphi U^{\leq k}\varphi,$$

where $a \in AP, \bowtie \in \{\leq, \geq\}$ and $\theta \in [0, 1]$. We refer to φ as a state formula and ρ as a path formula.

φ is interpreted in a state of the DTMC and it evaluates to either true or false. In particular, the symbol tt represents "true" and evaluates to true in every state. $\neg$ and $\wedge$ represent the negation and the conjunction boolean operators, with standard meaning. ϱ on the other hand is interpreted on a path, wherein, $X\varphi$ specifies that φ is true in the "next" state of the path and $\varphi_1 U^{\leq k}\varphi_2$ states that φ_2 is true within the next k states, and φ_1 is true in every state before φ_2 holds. $Pr_{\bowtie\theta}(\varrho)$ is true in a state if the probability of the set of all paths starting from that state which satisfy ϱ is $\bowtie \theta$, where $\bowtie \in \{\leq, \geq\}$. We will use a derived operator F in the sequel, that represents a property being satisfied in the future. More precisely $F^{\leq k}\varphi$ is a path formula that states that φ holds in some state within the next k states along the path, and can be expressed using the until operator U as $F^{\leq k}\varphi = ttU^{\leq k}\varphi$. Next, we formalize the semantics of CSL.

Semantics Let us fix a DTMC $\mathcal{T}$. Given a state s of $\mathcal{T}$, and a CSL state formula φ, we use $s \models \varphi$, to denote the fact that s satisfies the property φ in the model $\mathcal{T}$. Similarly, given an infinite path σ of $\mathcal{T}$ and a CSL path formula ρ, we use $\sigma \models \rho$ to denote that the path σ satisfies the path formula ρ. We define the satisfaction of φ, denoted as either $s \models \varphi$ or $\sigma \models \varrho$, inductively as follows:

- $s \models tt$ for all $s \in \mathcal{S}$;
- $s \models a$ if and only if $a \in L(s)$;
- $s \models \neg\varphi$ if and only if $s \not\models \varphi$;
- $s \models \varphi_1 \wedge \varphi_2$ if and only if $s \models \varphi_1$ and $s \models \varphi_2$;
- $s \models Pr_{\bowtie\ \theta}(\varrho)$ if and only if $prob(\mathcal{C}_\rho) \bowtie \theta$;

- $\sigma \models X\varphi$ if and only if $\sigma[1] \models \varphi$;
- $\sigma \models \varphi_1 U^{\leq k} \varphi_2$ if and only if for some $j \leq k$, $\sigma[j] \models \varphi_2$ and for all $0 \leq i < j$, $\sigma[i] \models \varphi_1$;

Here, $s \not\models \varphi$ denotes that $s \models \varphi$ is not true, and $\mathcal{C}_\rho = \{\sigma \in \mathcal{P}aths(\mathcal{T}) \mid \sigma \models \varrho\}$. Note that $\mathcal{C}_\rho$ is a measurable set, since, the satisfaction of a path σ only depends on a finite prefix whose length is upper-bounded by the sum of the bounds on the U operators in the formula. Let $\mathcal{C}_{\rho,n}$ denote the set of all paths in $\mathcal{C}_\rho$ whose length is n.

3.3 Verification Problem

In this section, we define the verification problem for discrete-time Markov chain where the correctness specification is given by a continuous stochastic logic formula.

Problem. Given a DTMC $\mathcal{T} = (\mathcal{S}, \longrightarrow, AP, L)$, a state $s \in \mathcal{S}$, and a CSL state formula φ, check whether the formula φ holds at state s, that is, $s \models \varphi$?

3.4 Illustrative Example

We illustrate DTMC with an example of a grid world robot navigation scenario.

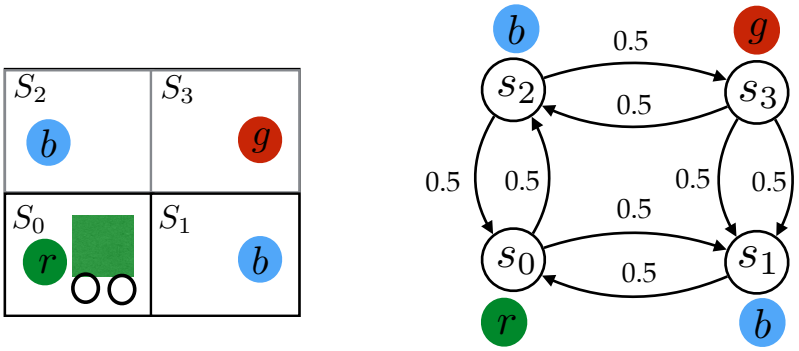

Figure 3.1: *Grid World Robot Navigation Scenario and the DTMC*

Consider a work-floor divided into a grid of size 2×2, consisting of four cells in which a robot is navigating as shown in Figure 3.1 on the left. The initial position of the robot

is in the cell labelled r; the label b indicates those cells where a robot can communicate with a base station; and the label g shows the goal/target cell. In each cell, the robot can move to each of its two neighboring cells (with which it shares a boundary line) with equal probability.

DTMC Modeling The robot behavior can be modelled as a DTMC with four states, s_0, s_1, s_2, and s_3, each corresponding to a cell, as shown in Figure 3.1 on the right. The transition relation is given by $\longrightarrow = \{(s_0, \rho_0), (s_1, \rho_1), (s_2, \rho_2), (s_3, \rho_3)\}$, where $\rho_0(s_1) = \rho_0(s_2) = \rho_1(s_0) = \rho_1(s_3) = \rho_2(s_0) = \rho_2(s_3) = \rho_3(s_1) = \rho_3(s_2) = 0.5$, and $\rho_i(s_j) = 0$, otherwise. We consider the set of atomic propositions to be $AP = \{r, b, g\}$, and the labelling function given by $L(s_0) = r$, $L(s_1) = L(s_2) = b$, and $L(s_3) = g$.

Probability Space Consider a set $\mathcal{C}$ of infinite paths starting from s_0 that reach s_3 at some point while avoiding s_2 until reaching s_3. The paths in $\mathcal{C}$ have prefixes of the form $(s_0 s_1)^+ s_3$. So, $prob(\mathcal{C}) = prob(s_0 s_1 s_3) + prob(s_0 s_1 s_0 s_1 s_3) + prob(s_0 s_1 s_0 s_1 s_0 s_1 s_3) + \ldots$, that is, $prob(\mathcal{C}) = (0.5)0.5 + (0.5)^3 0.5 + (0.5)^5 0.5 + \ldots = (0.5)0.5[1 + (0.5)^2 + (0.5)^4 + \ldots] = 0.25[\frac{1}{1-0.25}] = \frac{1}{3}$.

CSL Property We are interested in checking whether the robot starting from the initial cell r reaches the goal cell g within 4 steps with probability at least 0.1 while maintaining a probability of at least 0.2 to be able to communicate with the base station (which is enabled in the cells labelled b) in at most 2 steps. The corresponding CSL formula is given by:

$$\varphi = Pr_{\geq 0.1}[(Pr_{\geq 0.2} F^{\leq 2}\ b)\ U^{\leq 4}\ g] \tag{3.1}$$

Note that $\varphi = Pr_{\geq 0.1}(\varrho)$ where $\varrho = \psi\ U^{\leq 4}\ g$ and $\psi = (Pr_{\geq 0.2} F^{\leq 2}\ b)$. Note that ψ is true in every state, because, there is a path from every state whose probability is greater than 0.2 that reaches a state labelled b within 2 steps. More precisely, ψ is trivially true at s_1 and s_2, because b holds in these states. All paths from s_0, that is, those that start with $s_0 s_2$ and $s_0 s_1$, satisfy $F^{\leq 2}\ b$, and hence, s_0 satisfies $F^{\leq 2}\ b$ with probability 1. The DTMC satisfies the

formula φ at state s_0 because the following paths $\sigma_1 = s_0 \to s_1 \to s_3$ and $\sigma_2 = s_0 \to s_2 \to s_3$ satisfy the path formula ϱ, and $prob(\sigma_1) + prob(\sigma_2) = 0.25 + 0.25 = 0.50$, which is greater than 0.1. Note that $\sigma_1 \models \varrho$, because $\sigma_1[2] \models g$ and $\sigma_1[0]$ and $\sigma_1[1]$ satisfy ψ.

Chapter 4

Bayesian Hypothesis Testing

Our broad approach to verification is a statistical model-checking algorithm. To provide some background, in this section, we introduce statistical hypothesis testing and the Bayesian approach for testing a certain hypothesis.

4.1 A brief introduction to hypothesis testing

A hypothesis test aims to obtain some inference on the parameters of a probability distribution using some statistical tests. More precisely, let Y be a random variable whose probability distribution function (pdf) or probability mass function (pmf) depends on a parameter Θ. We intend to test whether Θ is above or below a certain value θ_0 called the threshold. Hence, we have two hypotheses:

$$H_0 : \Theta \geq \theta_0 \quad \text{vs} \quad H_1 : \Theta < \theta_0$$

where H_0 is referred to as the null hypothesis and H_1 as the alternate hypothesis. A statistical test typically consists of sampling from the distribution of Y to obtain a sample $\mathbf{y}$, computing a statistic which is some function of the sampled data $\mathbf{y}$, and then determining whether to accept or reject the null hypothesis H_0 based on the comparison of the statistic to a certain value. The value for comparison is chosen such that the outcome (accept/reject) has certain guarantees α and β on the probability of errors. More precisely, one needs to guarantee that the Type I error, namely, the probability that the test rejects the null hypothesis when it

is true, is less than α, and the Type II error, namely, the probability that the test fails to reject the null hypothesis when it is false, is less than β.

There are broadly two approaches to hypothesis testing, namely, a frequentist approach, wherein the parameter Θ is assumed to be a fixed unknown value, and the Bayesian approach, where Θ is considered as a random variable with a known distribution. In the paper, our objective is to develop a Bayesian test for solving the verification problem, hence, next, we present the details of the Bayesian approach.

4.2 Bayesian Hypothesis Testing

We present the Bayesian test for a Bernoulli distribution parameter Θ as proposed in[4;5]. Let Y be a Bernoulli random variable with parameter Θ, that is, Y is the outcome of a random experiment consisting of tossing a biased coin where the probability of head showing up is Θ. Let $P_{Y|\Theta}(y|\theta)$ be the probability mass function of Y, where $y = 0$ or 1, $0 \leq \theta \leq 1$, and $P_{Y|\Theta}(y|\theta)$ is the probability of head appearing in the coin toss for $y = 1$ and tail appearing for $y = 0$. In the Bayesian approach, the parameter Θ is assumed to be a random variable with probability density function $f_\Theta(\theta)$, which is also referred to as the prior distribution. The prior distribution is our belief on the true Θ before sampling.

Let $\mathbf{y} = (y_1, \cdots, y_n)$ be a random sample from Y. We will use $P_{Y|\Theta}(\mathbf{y}|\theta)$ to denote the probability of the sample $\mathbf{y}$ occurring as a result of n independent and identically distributed random Bernoulli trials. Let us consider the hypotheses, where θ_0 is the threshold:

$$H_0 : \Theta \geq \theta_0 \quad \text{vs} \quad H_1 : \Theta < \theta_0$$

The Bayes' test is based on the Bayes' factor, $B_Y(\mathbf{y}, \theta_0)$, as the statistic, which is the ratio of the probability of observing data $\mathbf{y}$, given that H_0 is true, denoted $f(\mathbf{y}|H_0)$, to the probability of observing data $\mathbf{y}$, given that H_1 is true, denoted $f(\mathbf{y}|H_1)$. Hence, the Bayes' factor is given by:

$$B_Y(\mathbf{y}, \theta_0) = \frac{f(\mathbf{y}|H_0)}{f(\mathbf{y}|H_1)} = \frac{\int_{\theta_0}^{1} P_{Y|\Theta}(\mathbf{y}|\theta) f_\Theta(\theta) d\theta}{\int_{0}^{\theta_0} P_{Y|\Theta}(\mathbf{y}|\theta) f_\Theta(\theta) d\theta} * \frac{P_1}{P_0},$$

$$P_0 = \int_{\theta_0}^{1} f_\Theta(\theta)d\theta, P_1 = \int_{0}^{\theta_0} f_\Theta(\theta)d\theta.$$

The next theorem from[4;5] provides a bound on the Type I and Type II errors for the hypotheses H_0 and H_1 on the parameter Θ of the Bernoulli random variable Y based on Bayes' factor.

Theorem 1 (Bayes' test)**.** *Consider a Bernoulli random variable Y with parameter Θ and threshold θ_0. Let $H_0 : \Theta \geq \theta_0$ and $H_1 : \Theta < \theta_0$ be the null and alternate hypotheses, respectively. Consider the test that accepts H_0 when $B_Y(\mathbf{y}, \theta_0) \geq \frac{1}{\beta}$ and rejects H_0 when $B_Y(\mathbf{y}, \theta_0) \leq \alpha$. Then α and β are the upper bounds of Type I error and Type II error, respectively.*

Note that for a certain sample $\mathbf{y}$, Bayes' test accepts or rejects the null hypothesis only if the Bayes' factor is in the range $[\frac{1}{\beta}, \alpha]$. Hence, for given Type I and Type II errors α and β, the algorithm based on Bayes' test consists of iteratively sampling from Y until the test statistic, Bayes' factor B_Y, computed from the sample values falls outside the range $[\frac{1}{\beta}, \alpha]$.

Computation of B_Y Bayes' factor for Y given $\mathbf{y}$ and θ_0 is given by:

$$B_Y(\mathbf{y}, \theta_0) = \frac{f(\mathbf{y}|H_0)}{f(\mathbf{y}|H_1)} = \frac{\int_{\theta_0}^{1} P_{Y|\Theta}(\mathbf{y}|\theta) f_\Theta(\theta)d\theta}{\int_{0}^{\theta_0} P_{Y|\Theta}(\mathbf{y}|\theta) f_\Theta(\theta)d\theta} * \frac{P_1}{P_0}$$

where $P_0 = \int_{\theta_0}^{1} f_\Theta(\theta)d\theta$, $P_1 = \int_{0}^{\theta_0} f_\Theta(\theta)d\theta$, and $f_\Theta(\theta)$ is a prior distribution of Y. Given that $Y|\Theta$ is a Bernoulli random variable with parameter Θ, B_Y can be computed using the following equation:

$$B_Y(\mathbf{y}, \theta_0) = \frac{P_1}{P_0} * \frac{\int_{\theta_0}^{1} (\theta)^{(m)}(1-\theta)^{(n-m)} f_\Theta(\theta)d\theta}{\int_{0}^{\theta_0} \theta^{(m)}(1-\theta)^{(n-m)} f_\Theta(\theta)d\theta}, \text{ where}$$

$P_1 = \int_{0}^{\theta_0} f_\Theta(\theta)d\theta, P_1 = \int_{0}^{\theta_0} f_\Theta(\theta)d\theta$, $m = \sum_{i=1}^{n} y_i$ and n is the sample size.

Next, we briefly discuss the choice of the prior distribution $f_\Theta(\theta)$ for the parameter Θ of the Bernoulli distribution. Note that Θ ranges in the interval $[0, 1]$, hence, we need the support of f_Θ to be $[0, 1]$. We assume a Beta distribution as a prior for Θ whose support is

$[0, 1]$, and whose probability density function consists of two shape parameters a and b.

Beta distributions are a general class of distributions that can capture a wide range of distributions by setting different values for the parameters a and b. For instance, when $a = b = 1$, $f_\Theta(\theta) = \frac{1}{Beta(a,b)} = 1$ for $0 \leq \theta \leq 1$, since $Beta(1, 1) = 1$, that is, we obtain a uniform distribution. Further, Beta prior is the 'conjugate' prior of Bernoulli distribution in the sense that the prior and posterior distributions belong to the same family of distributions. This simplifies some of the computation as we will see in the next section.

Chapter 5

Bayesian SMC for Non-nested CSL Verification

In this section, we formulate the verification problem as a hypothesis testing problem. The nested probabilistic operators pose a challenge to statistical analysis, so, we start with formulae without nested probabilistic operators, and then extend our analysis to nested operators.

5.1 Verifying non-nested formulas

Verification problem can be formulated as a hypothesis testing problem. First, we consider a formula with a single probabilistic operator at the top, namely, $\varphi = Pr_{\geq\theta_0}(\varrho)$, where ϱ is a path formula and does not contain any probabilistic operator Pr within it. We refer to such formulae as *non-nested* formulae.

The satisfaction of the path formula ϱ by a path σ depends only on a finite prefix of it, whose length, say L_ϱ, can be upper bounded, for instance, by the sum of the k's in $U^{\leq k}$ operators appearing within ϱ. Hence, the satisfaction of φ is a state s can be determined by considering all paths from s of length L_ϱ, verifying if the path satisfies ϱ, summing up the probabilities of all the paths that satisfy ϱ and checking if it is $\geq \theta_0$. Note that whether $\sigma \models \varrho$ can be computed in time linear in the L_ϱ and size of ϱ. While the above approach

provides a deterministic procedure, it is highly inefficient because we need to enumerate all paths of length L_ϱ, which are exponentially many in L_ϱ.

To avoid the exponential blow-up in verification complexity, we turn to a statistical approach wherein the verification problem is formulated as a hypothesis testing problem as follows. Let us consider a sample space $\mathcal{C}$ consisting of all finite paths of $\mathcal{T}$ starting from s up to a given length L_ϱ, and let $\mathcal{C}_\varrho$ be the set of paths in $\mathcal{C}$ which satisfy ϱ. Recall that the probability associated with a finite path σ is given by $prob(\sigma)$. A standard random walk on the DTMC will sample from $\mathcal{C}$ such that σ appears with probability $prob(\sigma)$. Let Y be a random variable on this space, such that $Y(\sigma) = 1$ if $\sigma \models \varrho$, and 0, otherwise. Note that $P[Y = 1] = prob(\mathcal{C}_\varrho)$. Hence, Y is a Bernoulli random variable with parameter $\Theta = prob(\mathcal{C}_\varrho)$.

Further, to solve the verification problem, does $s \models \varphi = Pr_{\geq\theta_0}(\varrho)$, we don't need the exact value of Θ, but we need to know whether it is greater than or equal to θ_0. This is essentially a hypothesis test, where the null hypothesis is given by $H_0 : \Theta \geq \theta_0$ vs the alternative hypothesis $H_1 : \Theta < \theta_0$. We can use Bayes' test to check if $s \models \varphi$, that is, $s \models \varphi$ if H_0 is accepted by the Bayes' test and $s \not\models \varphi$ if H_1 is accepted by the Bayes' test. Theorem 1 provides upper bounds on Type 1 and Type II errors. The Type I error corresponds to the probability that the test falsely concludes that s does not satisfy φ, and type II error to the probability that the test falsely concludes that s satisfies φ. Unlike the standard model-checking algorithms[40], where we obtain with probability 1 whether $s \models \varphi$, here, we can only say with high probability whether $s \models \varphi$. However, as we will see in the experimental section, statistical methods are much faster than traditional state space exploration based model-checking algorithms.

Our algorithm for the non-nested case is summarized in Algorithm 1. Algorithms checks if $\varphi = Pr_{\bowtie\theta_0}(\rho)$ is satisfied in a state s of $\mathcal{T}$, under the assumption that the prior distribution of Θ is a Beta distribution with parameters a and b. Our algorithm is a randomized algorithm which outputs the correct answer to $s \models \varphi$ with high probability, that is, the Type I and II error probabilities are bounded by α and β. The algorithm consists of sampling a sequence of paths σ of length $len(\varrho)$ from $\mathcal{T}$ starting from s and checking if $\sigma \models \varrho$. Here, n counts the total number of samples and m counts those that satisfy ϱ. BayesFactor(θ_0, n, m, a,

Algorithm 1: Bayes_SMC_Base - SMC of a non-nested CSL formula

Input: s - state, $\mathcal{T}$ - DTMC, $\varphi = Pr_{\bowtie\theta_0}(\rho)$ - non-nested CSL formula, a, b - parameters for beta prior, α, β - bound on Type I and Type II errors

Output: Answer if $s \models \varphi$ in $\mathcal{T}$ with confidence α, β

```
1  begin
2      Set n = 0, m = 0
3      Set L_ϱ to be the depth of ϱ
4      while True do
5          Increment n
6          Generate a (random) path σ from state s in T of length L_ϱ
7          if σ satisfies ρ then
8              Increment m
9          Set B = BayesFactor(θ_0, n, m, a, b)
10         if B > 1/β then
11             return true
12         if B < α then
13             return false
```

b) computes the B as given below, based on which the decision to accept or reject is made to ensure Type I and Type II errors are within α and β, respectively, in accordance with Theorem 1.

$$B = \frac{P_1}{P_0} * \frac{\int_{\theta_0}^{1}(\theta)^m(1-\theta)^{(n-m)}f_\Theta(\theta))d\theta}{\int_{0}^{\theta_0}(\theta)^m(1-\theta)^{(n-m)}f_\Theta(\theta)d\theta}, \quad \text{where } P_0 = \int_{\theta_0}^{1} f_\Theta(\theta)d\theta, P_1 = \int_{0}^{\theta_0} f_\Theta(\theta)d\theta$$

We can summarize the correctness criterion of Algorithm 1 as follows:

Theorem 2. *Let s, $\mathcal{T}$, φ, a, b, α, and β, be as in Algorithm 1. If $s \models \varphi$ in $\mathcal{T}$, then Algorithm 1 outputs true with probability at least $(1-\alpha)$. If $s \not\models \varphi$ in $\mathcal{T}$, then Algorithm 1 outputs false with probability at least $(1-\frac{1}{\beta})$.*

Chapter 6

Bayesian SMC for Nested CSL

In this section, we extend the Bayesian approach to nested probabilistic operators. Bayesian SMC has been explored previously for non-nested operators[4;5]. A frequentist SMC approach based on acceptance sampling has been explored for the complete CSL logic[3]. Here, we build upon ideas from these papers to address the problem of Bayesian SMC for the complete CSL logic.

6.1 Overview of the approach

Standard model-checking algorithms for verifying general CSL formulae work bottom-up, where the satisfaction of state subformulae are evaluated at each state, and the satisfaction of a formula containing these subformulae are inferred from it. Our broad approach is a similar recursive algorithm, however, there are several intricacies due to the fact that we can compute the satisfaction of subformulae exactly using an SMC algorithm.

Let us consider a CSL formula $\varphi = Pr_{\geq\theta_0}(\varrho)$, where the path formula ϱ could potentially have probabilistic operators Pr. We define a top-level formula of ϱ, a subformula of ϱ of the form $Pr_{\geq\theta_0}(\varrho')$ which are not contained inside any probabilistic operators within ϱ. For instance, consider $\varphi = Pr_{\geq 0.1}(\varrho)$ where $\varrho = \psi\ U^{\leq 4}\ g$ and $\psi = (Pr_{\geq 0.2}F^{\leq 2}\ b)$ as shown in Figure 6.1. ψ is a top-level subformula of ϱ. Suppose $\psi = (Pr_{\geq 0.2}F^{\leq 2}\ \psi')$, where

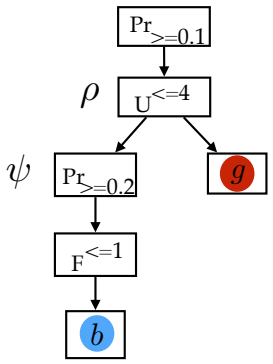

Figure 6.1: $Pr_{\geq 0.1}(Pr_{\geq 0.2}(F^{\leq 1}b)U^{\leq 4}g)$

$\psi' = Pr_{\geq 0.5}(Xg)$. Then ψ' would not be a top-level subformula of ϱ, since it is contained within another subformula with a probabilistic operator at the root, that is, ψ, which is a subformula of ϱ.

Suppose that we have deduced the satisfaction of all top-level formulas of ϱ in each state of the DTMC, that is, whether $s \models \varphi'$ for each state s of the DTMC and each top-level formula φ' of ϱ. Then the verification of $\varphi = Pr_{\geq \theta_0}(\varrho)$ could be performed as in the case of the non-nested operators. That is, given any path σ, we can define a random variable Y such that $Y(\sigma) = 1$ if the σ satisfies ϱ, which can be deduced (computed) deterministically if the values of $s \models \varphi'$ for each state s and top-level formula φ' of ϱ are known.

Again, consider the formula φ from Figure 6.1. The satisfaction of ϱ in a path σ depends on the satisfaction of ψ along states in ϱ. Suppose that we have checked for each state of the DTMC, if ψ holds in that state. We can then use Algorithm 1 to check if $s \models \varphi$. More precisely, we can verify the non-nested formula $\varphi = Pr_{\geq 0.1}(p_\psi \; U^{\leq 4} \; g)$, where the labels of DTMC now consist of an additional label p_ψ which is added to the label of exactly those sets that satisfy ψ. Hence, we could verify a nested formula in a bottom up fashion, provided we can compute the satisfaction of its top-level subformulae exactly.

However, our objective is to design a statistical model-checking algorithm for verifying CSL formulae, and avoid computationally expensive probabilistic model-checking algorithms. Hence, we will recursively call SMC algorithms on top-level formulae ψ of ϱ, and use the

result in verifying the satisfaction of ϱ. In this case, the answer to $s \models \psi$ is not definitive, but has some error given by Type I or Type II errors. Hence, we need to factor in this uncertain answer for the top-level subformulae in the analysis of ϱ. We consider a random variable Z such that $Z(\sigma) = 1$ if σ satisfies ϱ using the values returned by SMC for $s \models \psi$. The values of $Z(\sigma)$ and $Y(\sigma)$ could differ, since, SMC could answer the question $s \models \psi$ incorrectly. However, we can bound probability with which Z differs from Y. This enables us to approximate the Bayes' test based on Y, through a Bayes' test based on Z, and use the latter for Bayesian statistical test for verifying φ.

Our broad approach consists of the following steps.

1. **Recursive step:** Compute recursively the $s \models \psi$ for all states s of the DTMC $\mathcal{T}$ and top level formulas ψ of ϱ using the Bayesian SMC algorithm such that given Type I and Type II error bounds, α_ψ and β_ψ, are achieved, respectively. (We assume that the bounds are the same for all states, this can be easily relaxed).

2. **Error Propagation:** Propagate the Type I and Type II error bounds, α_ψ and β_ψ for the top level formulas ψ of ϱ to the root of the formula ϱ to obtain α_ϱ and β_ϱ. α_ϱ and β_ϱ provide bounds on the Type I and Type II errors, Z with respect to Y, that is, the probability that $\sigma \models \varrho$ is answered incorrectly when SMC results are used for satisfaction of ψ in a state as compared to when exact satisfaction answers are used. Formally, α_ϱ and β_ϱ computed satisfy $P[Z = 0|Y = 1] \leq \alpha_\varrho$ and $P[Z = 1|Y = 0] \leq \beta_\varrho$.

3. **Approximate Bayesian Test:** While Bayes' test deduces whether $s \models \varphi$ by using the values of samples from Y, we can only sample values of Z using our recursive statistical approach. Hence, we provide an approximate Bayesian test, that provides inferences on $s \models \varphi$ with given Type I and Type II error guarantees using samples of Z.

In Sections 6.2 and 6.3, we provide the details of the error propagation step and the approximate Bayesian test, respectively. In Section **??**, we present the algorithmic details and illustrate on an example.

6.2 Error Propagation

In this section, we describe an inductive definition to compute the Type I and Type II errors, $E_1(\varrho)$ and $E_2(\varrho)$, respectively, corresponding to satisfaction of ϱ (in any path σ of DTMC $\mathcal{T}$) given the Type I and Type II errors for the SMC of top-level formulae ψ of ϱ, denoted $E_1(\psi)$ and $E_2(\psi)$, respectively. In particular, we provide $E_1(\varphi)$ and $E_2(\varphi)$, for the case $\varphi = tt, a, \neg\psi, \psi_1 \wedge \psi_2$ and $\varrho = X\psi, \psi_1 U^{\leq k} \psi_2$.

- $E_1(tt) = E_2(tt) = 0$;
- $E_1(a) = E_2(a) = 0$;
- $E_1(\neg\psi) = E_2(\psi)$, $E_2(\neg\psi) = E_1(\psi)$;
- $E_1(\psi_1 \wedge \psi_2) = E_1(\psi_1) + E_1(\psi_2)$, $E_2(\psi_1 \wedge \psi_2) = \max\{E_2(\psi_1), E_2(\psi_2)\}$;
- $E_1(X\psi) = E_1(\psi)$, $E_2(X\psi) = E_2(\psi)$;
- $E_1(\psi_1 U^{\leq k} \psi_2) = kE_1(\psi_1) + E_1(\psi_2)$, $E_2(\psi_1 U^{\leq k} \psi_2) = (k+1)\max\{E_2(\psi_1), E_2(\psi_2)\}$.

Below we provide a sketch of correctness of the rules. Note that the Type I error $E_1(\neg\psi)$ is the probability that the statistical evaluation of $\neg\psi$ returns false when it is true. But this is equivalent to the probability of the statistical evaluation of φ returning true when it is false. Hence, we obtain $E_1(\neg\psi) = E_2(\psi)$. A similar argument provides $E_2(\neg\psi) = E_1(\psi)$.

$E_1(\psi_1 \wedge \psi_2)$ is the probability that the statistical evaluation of $\psi_1 \wedge \psi_2$ returns false when it is true. That is, ψ_1 and ψ_2 are true but either the statistical evaluation of ψ_1 says false or that of ψ_2 says false. This probability can be upper bounded by the sum of Type I errors for ψ_1 and ψ_2. $E_2(\psi_1 \wedge \psi_2)$ is the probability that the statistical evaluation of $\psi_1 \wedge \psi_2$ returns true when it is false. There are three cases here: ψ_1 is true and ψ_2 is false, ψ_1 is false and ψ_2 is true, and ψ_1 and ψ_2 are both false. Let us consider the case where ψ_1 is true and ψ_2 is false, ψ_1 is false. The statistical evaluation of $\psi_1 \wedge \psi_2$ returns true when the statistical evaluation of ψ_1 and that of ψ_2 both return true. The probability that the statistical evaluation of ψ_1 returns true, when ψ_1 is true (and ψ_2 is false), is upper bounded by 1, while the probability

that the statistical evaluation of ψ_2 returns true, when ψ_2 is false (and ψ_1 is true) is upper bounded by the Type II error of ψ_2. Hence, the probability that the statistical evaluation of $\psi_1 \wedge \psi_2$ returns true, when ψ_1 is true and ψ_2 is false, is the product of the above, namely, $1 \times E_2(\psi_2) = E_2(\psi_2)$. Similarly, for the other two cases we can upper bound the probability by $E_2(\psi_1)$ and $E_2(\psi_1) \times E_2(\psi_2)$. The overall upper bound is the maximum of all three, namely, $\max\{E_2(\psi_1), E_2(\psi_2)\}$.

Note that $X\psi$ is satisfied by a path σ exactly when ψ is satisfied at $\sigma[1]$. Hence, the statistical evaluation of $X\psi$ for σ is incorrect exactly when that of ψ is incorrect at $\sigma[1]$. Therefore the error probabilities carry over. Finally, the Type I and Type II errors for $\psi_1 U^{\leq k} \psi_2$ are obtained by interpreting $\psi_1 U^{\leq k} \psi_2$ as the equivalent formula $\neg \bigwedge_{i=0}^{k} \neg(\bigwedge_{j=0}^{i-1} X^j \psi_1 \wedge X^i \psi_2)$, and using the previous rules.

6.3 Approximate Bayesian Test

In this section, we provide Bayes' test for the verification of a CSL formula $\varphi = Pr_{\geq \theta_0}(\varrho)$ on a DTMC $\mathcal{T}$ at a state s, where ϱ could potentially have probabilistic operators. As in the non-nested case, define a random variable Y, where $Y(\sigma) = 1$ if σ satisfies ϱ, and 0 otherwise. Note that Y is a Bernoulli random variable with parameter $\Theta = prob(\mathcal{C}_\varrho)$, where $\mathcal{C}_\varrho$ is the set of paths of $\mathcal{T}$ which satisfy ϱ. For the verification of φ, we need to test the following hypothesis:

$$H_0 : \Theta \geq \theta_0 \quad \text{vs} \quad H_1 : \Theta < \theta_0$$

Ideally, we would like to compute the Bayes' factor B_Y over sampled data $\mathbf{y}$ consisting of paths starting from the state s and use that in checking whether the state s satisfies φ. However, given a path σ, we cannot check if it satisfies ϱ exactly, because we do not know the truth values of the nested probabilistic operator within ϱ at the different states. Instead, we have access to an SMC algorithm that deduces the satisfaction of nested probabilistic operators in the states of DTMC with certain error bounds. In Section 6.2, we provided the inductive definition to propagate these errors to ϱ. Hence, we have access to a random variable Z such that $Z(\sigma)$ is 1 if σ satisfies ϱ when SMC values are used for top-level

formulae. Note that Z will not coincide with Y for a given σ always, but will agree with high probability. More precisely, we have:

$$P[Z = 0|Y = 1] \leq \alpha_\varrho \quad P[Z = 1|Y = 0] \leq \beta_\varrho \tag{6.1}$$

Moreover, Z can be interpreted as a Bernoulli random variable with some parameter Θ'. Note that in the Bayesian approach, Θ itself is a random variable with a sample space Ω. Our next task is to bound the difference between Θ and Θ' (for every point in the sample space for Θ).

Proposition 1. *Let $Y \sim \mathcal{B}(\Theta)$ and $Z \sim \mathcal{B}(\Theta')$ be Bernoulli random variables with parameter Θ and Θ', respectively, and let Equation 6.1 hold. Then, we have $-\alpha_\varrho \leq \Theta' - \Theta \leq \beta_\varrho$.*

Proof. We know the following facts.

- $P[Y = 1] = \Theta, P[Y = 0] = 1 - \Theta$, $P[Z = 1] = \Theta', P[Z = 0] = 1 - \Theta'$ since Y and Z are Bernoulli random variables with parameters Θ and Θ', respectively;
- $P[Z = 0|Y = 1] \leq \alpha_\varrho, P[Z = 1|Y = 0] \leq \beta_\varrho$, from Equation 6.1, and
- $P[Z = 1|Y = 1] \leq 1, P[Z = 0|Y = 0] \leq 1$ from the properties of probability measures.

We can deduce:

- $\Theta' = P[Z = 1] = P[Z = 1|Y = 0]P[Y = 0] + P[Z = 1|Y = 1]P[Y = 1] \leq \beta_\varrho(1 - \Theta) + 1.\Theta$ which implies $\Theta' - \Theta \leq \beta_\varrho(1 - \Theta) \leq \beta_\varrho$.
- $1 - \Theta' = P[Z = 0] = P[Z = 0|Y = 0]P[Y = 0] + P[Z = 0|Y = 1]P[Y = 1] \leq 1.(1 - \Theta) + \alpha_\varrho\Theta$ which implies $\Theta' - \Theta \geq -\alpha_\varrho\Theta \geq -\alpha_\varrho$. □

Next, we present Bayes' test on Y using Bayes' factor for a test on Z as stated in the following theorem.

Theorem 3. *Let $Y \sim \mathcal{B}(\Theta)$ and $Z \sim \mathcal{B}(\Theta')$ be Bernoulli random variables with parameter Θ and Θ', respectively, and Equation 6.1 hold. Given hypothesis*

$$H_0 : \Theta \geq \theta_0 \quad vs \quad H_1 : \Theta < \theta_0$$

Consider the test that:

- *Accepts H_0 when $B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1$, and*
- *Rejects H_0 when $B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2$,*

where B_Z denotes the Bayes' factor for a test on Z, $r_1 = P[\Theta < \theta_0]/P[\Theta < \theta_0 + \alpha_\varrho + \beta_\varrho]$ and $r_2 = P[\Theta \geq \theta_0]/P[\Theta \geq \theta_0 - \alpha_\varrho - \beta_\varrho]$. Then α and β are the upper bounds of Type I error and Type II error, respectively.

Proof. Note that from Proposition 1, $\Theta(\omega) \geq \theta_0$ implies $\Theta'(\omega) \geq \Theta(\omega) - \alpha_\varrho \geq \theta_0 - \alpha_\varrho$. Hence, $\{\omega \,|\, \Theta(\omega) \geq \theta_0\} \subseteq \{\omega \,|\, \Theta'(\omega) \geq \theta_0 - \alpha_\varrho\}$. Similarly, $\Theta(\omega) < \theta_0$ implies $\Theta'(\omega) \leq \Theta(\omega) + \beta_\varrho < \theta_0 + \beta_\varrho$. Hence, $\{\omega \,|\, \Theta(\omega) < \theta_0\} \subseteq \{\omega \,|\, \Theta'(\omega) < \theta_0 + \beta_\varrho\}$.

Also, from Proposition 1, $\Theta(\omega) \geq \Theta'(\omega) - \beta_\varrho$. Hence, $\Theta'(\omega) \geq \theta_0 - \alpha_\varrho$ implies $\Theta(\omega) \geq \theta_0 - \alpha_\varrho - \beta_\varrho$. Similarly, from Proposition 1, $\Theta(\omega) \leq \Theta'(\omega) + \alpha_\varrho$. Hence, $\Theta'(\omega) < \theta_0 + \beta_\varrho$ implies $\Theta(\omega) < \theta_0 + \beta_\varrho + \alpha_\varrho$.

First, we show that α is a Type I error bound for the hypothesis test, that is, we show that $P(\{\text{reject } H_0\}|H_0) \leq \alpha$.

$$\begin{aligned}
&P(\{\text{reject } H_0\} \mid H_0) = P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2 \mid H_0) = P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2 \mid \Theta \geq \theta_0) \\
&= \frac{P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2, \Theta \geq \theta_0)}{P(\Theta \geq \theta_0)} \leq \frac{P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2, \Theta' \geq \theta_0 - \alpha_\varrho)}{P(\Theta \geq \theta_0)} \\
&= \frac{P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2, \Theta' \geq \theta_0 - \alpha_\varrho)}{P(\Theta' \geq \theta_0 - \alpha_\varrho)} \frac{P(\Theta' \geq \theta_0 - \alpha_\varrho)}{P(\Theta \geq \theta_0)} \\
&\leq P(B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \leq \alpha r_2 \,|\, \Theta' \geq \theta_0 - \alpha_\varrho) \frac{P(\Theta \geq \theta_0 - \alpha_\varrho - \beta_\varrho)}{P(\Theta \geq \theta_0)} \leq (\alpha r_2)/r_2 = \alpha
\end{aligned}$$

Next, we show that β is a Type II error bound for the hypothesis test, that is, we show that $P(\{\text{accept } H_0\}|H_1) \leq \beta$.

$$
\begin{aligned}
&P(\{\text{accept } H_0\} \mid H_1) = P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1 \mid H_1) = P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1 \mid \Theta < \theta_0) \\
&= \frac{P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1, \Theta < \theta_0)}{P(\Theta < \theta_0)} \leq \frac{P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1, \Theta' < \theta_0 + \beta_\varrho)}{P(\Theta < \theta_0)} \\
&= \frac{P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1, \Theta' < \theta_0 + \beta_\varrho)}{P(\Theta' < \theta_0 + \beta_\varrho)} \frac{P(\Theta' < \theta_0 + \beta_\varrho)}{P(\Theta < \theta_0)} \\
&\leq P(B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \geq 1/\beta r_1 \,|\, \Theta' < \theta_0 + \beta_\varrho) \frac{P(\Theta < \theta_0 + \beta_\varrho + \alpha_\varrho)}{P(\Theta < \theta_0)} \leq \beta r_1 / r_1 = \beta
\end{aligned}
$$

□

Note that $\alpha_\rho, \beta_\rho > 0$ implies $0 < r_1, r_2 < 1$, $B_Z(\mathbf{z}, \theta_0 + \beta_\varrho) \leq B_Z(\mathbf{z}, \theta_0)$ and $B_Z(\mathbf{z}, \theta_0 - \alpha_\varrho) \geq B_Z(\mathbf{z}, \theta_0)$. Hence, the approximate Bayes' test is stricter than the traditional Bayes' test for Z.

Computation of B_Z: Theorem 3 provides us a method to perform the hypothesis test on Θ using Z. This requires the computation of B_Z that involves the $P_{Z|\Theta'}$ which is a Binomial distribution with parameter Θ' and $f_{\Theta'}$, the prior distribution for Θ'. We do not know the exact distribution of Θ', however, we know that it is close to that of Θ. We use this fact to obtain upper and lower bounds on the values of B_Z and use that in the hypothesis test.

Bayes' factor for Z given $\mathbf{z}$ and θ_0' is given by:

$$
B_Z(\mathbf{z}, \theta_0') = \frac{\int_{\theta_0'}^{1} P_{Z|\Theta'}(\mathbf{z}|\theta') f_{\Theta'}(\theta') d\theta'}{\int_{0}^{\theta_0'} P_{Z|\Theta'}(\mathbf{z}|\theta') f_{\Theta'}(\theta') d\theta'} * \frac{P_1}{P_0}
$$

where $P_0 = \int_{\theta_0'}^{1} f_{\Theta'}(\theta') d\theta'$, $P_1 = \int_{0}^{\theta_0'} f_{\Theta'}(\theta') d\theta'$, and $f_{\Theta'}(\theta')$ is a prior distribution of Z. Note that exact computation of B_Z is not possible because prior distribution $f_{\Theta'}$ of Z is not known. Hence, we find the lower and upper bound on B_Z via lower and upper bound on the unknown prior distribution $f_{\Theta'}$.

Let us assume that Θ' is a function of Θ, say, $\Theta' = g(\Theta)$ such that $-\alpha_\varrho \leq g(\Theta) - \Theta \leq \beta_\varrho$.

Note that this is a stronger assumption than $-\alpha_\varrho \leq g(\Theta) - \Theta \leq \beta_\varrho$, however, a reasonable assumption given this fact. (Our assumption requires that an ω that map to θ through Θ maps to $g(\theta)$ through Θ'.) Further, let us assume that g^{-1} is differentiable and its derivative is bounded by some constants, $c \leq \frac{d}{d\theta'} g^{-1}(\theta') \leq d$ for all $\theta' \in [0, 1]$. Now, we can use transformation of random variables to obtain the pdf $f_{\Theta'}$ of Θ'.

$$f_{\Theta'}(\theta') = f_\Theta(g^{-1}(\theta')) \frac{d}{d\theta'} g^{-1}(\theta'),$$

and bound it using the derivate bounds:

$$c f_\Theta(g^{-1}(\theta')) \leq f_{\Theta'}(\theta') \leq d f_\Theta(g^{-1}(\theta')),$$

since, from our assumption, $\frac{d}{d\theta'} g^{-1}(\theta') \in [c, d]$. Plugging $\Theta = g^{-1}(\theta')$ in $-\alpha_\varrho \leq g(\Theta) - \Theta \leq \beta_\varrho$, we get $-\alpha_\varrho \theta' - g^{-1}(\theta') \leq \beta_\varrho$. We can conclude $\|\theta' - g^{-1}(\theta')\| \leq \max\{\alpha_\varrho, \beta_\varrho\}$.

Suppose f_Θ is Lipschitz continuous with Lipschitz constant L, that is, $\|f_\Theta(\theta_1) - f_\Theta(\theta_2)\| \leq L\|\theta_1 - \theta_2\|$. Then, $\|f_\Theta(\theta') - f_\Theta(g^{-1}(\theta'))\| \leq L\|\theta' - g^{-1}(\theta')\| \leq L \max\{\alpha_\varrho, \beta_\varrho\}$. Hence, $f_\Theta(\theta') - L \max\{\alpha_\varrho, \beta_\varrho\} \leq f_\Theta(g^{-1}(\theta')) \leq f_\Theta(\theta') + L \max\{\alpha_\varrho, \beta_\varrho\}$. Therefore,

$$c(f_\Theta(\theta') - L \max\{\alpha_\varrho, \beta_\varrho\}) \leq f_{\Theta'}(\theta') \leq d(f_\Theta(\theta') + L \max\{\alpha_\varrho, \beta_\varrho\})$$

We can compute upper and lower bounds on $B_Z(\mathbf{z}, \theta')$ as in the next proposition.

Proposition 2. *Let $Y \sim \mathcal{B}(\Theta)$ and $Z \sim \mathcal{B}(\Theta')$ be Bernoulli random variables with parameter Θ and Θ', respectively, and let Equation 6.1 hold. Assume $Z = g(Y)$, where g^{-1} is differentiable and in the range $[c, d]$. Let f_Θ be Lipschitz continuous with Lipschitz constant L. Let $\gamma = L \max\{\alpha_\varrho, \beta_\varrho\}$. Let*

$$B_Z(\mathbf{z}, \theta'_0)_{min} = \frac{c^2}{d^2} * \frac{P_1^{min}}{P_0^{max}} * \frac{\int_{\theta'_0}^1 (\theta')^m (1-\theta')^{(n-m)} (f_\Theta(\theta') - \gamma) d\theta'}{\int_0^{\theta'_0} (\theta')^m (1-\theta')^{(n-m)} (f_\Theta(\theta') + \gamma) d\theta'}$$

$$B_Z(\mathbf{z},\theta_0')_{max} = \frac{d^2}{c^2} * \frac{P_1^{max}}{P_0^{min}} * \frac{\int_{\theta_0'}^1 (\theta')^{(m)}(1-\theta')^{(n-m)}(f_\Theta(\theta')+\gamma)d\theta'}{\int_0^{\theta_0'} \theta'^{(m)}(1-\theta')^{(n-m)}(f_\Theta(\theta')-\gamma)d\theta'}, \text{ where}$$

$$P_0^{min} = \int_{\theta_0'}^1 (f_\Theta(\theta')-\gamma)d\theta', P_0^{max} = \int_{\theta_0'}^1 (f_\Theta(\theta')+\gamma)d\theta',$$

$$P_1^{min} = \int_0^{\theta_0'} (f_\Theta(\theta')-\gamma)d\theta', P_1^{max} = \int_0^{\theta_0'} (f_\Theta(\theta')+\gamma)d\theta',$$

$m = \sum_{i=1}^n z_i$ and n is the sample size of $\mathbf{z}$. Then, we have

$$B_Z(\mathbf{z},\theta_0')_{min} \leq B_Z(\mathbf{z},\theta_0') \leq B_Z(\mathbf{z},\theta_0')_{max}.$$

The following proposition states that Beta distribution is a Lipschitz continuous function.

Proposition 3. *Let f_Θ be a Beta prior distribution for fixed shape parameters a, b, that is, $f_\Theta(\theta) = \frac{\theta^a(1-\theta)^b}{Beta(a,b)}$, for $0 \leq \theta \leq 1$. Then f_Θ is a Lipschitz continuous function with Lipschitz constant $L = \frac{(a+b)}{Beta(a,b)}$.*

6.4 Bayesian SMC Algorithm for Nested CSL

Algorithm 2 describes the verification of a CSL formula $\varphi = Pr_{\bowtie\theta_0}(\rho)$, where ρ could potentially have nested probabilistic operators. If ρ is non-nested, then the algorithm calls the Bayesian SMC for the non-nested case. If ρ is a nested formula, $\alpha_\varrho, \beta_\varrho \in (0,1)$ are chosen such that $\theta_0 - \alpha_\varrho, \theta_0 + \beta_\varrho \in (0,1)$. We leave the exact procedure to be figured out during the implementation, as it does not affect the correctness of the algorithm.

Comp_Top_Error(ϱ,α_ϱ, β_ϱ) returns (a vector of) error bounds α_ψ, β_ψ for all top-level formula ψ such that $\alpha_\varrho, \beta_\varrho$ are upper bounds on the errors propagated from the top-level formulae ψ to ρ using the error propagation rules in Section 6.2. The Bayesian SMC algorithm is called recursively on each top-level formula ψ and state s' of the DTMC $\mathcal{T}$ to obtain the "truth" values $t_{\psi,s'}$ with guaranteed error bounds α_ψ, β_ψ.

ρ is then treated as a "non-nested" formula where each top level formula ψ is treated as a proposition with truth value $t_{\psi,s'}$ in state s'. Let L_ρ be the depth of this "non-nested"

formula. Samples σ of length L_ρ are generated, and $\sigma \models \rho$ is checked using the computed values $\{t_{\psi,s'}\}$. n is the total number of samples and m the number of samples that satisfy ρ as above. BayesFactorMin and BayesFactorMax implement $B_Z(\mathbf{z}, \theta')_{min}$ and $B_Z(\mathbf{z}, \theta')_{max}$, respectively. ProbBounds computes r_1 and r_2 from Theorem 3; lower and upper bounds on B_Z are used to check the acceptance/rejection conditions in the theorem.

Algorithm 2: Bayes_SMC_Nested - SMC of a non-nested CSL formula

Input: s - state, $\mathcal{T}$ - DTMC, $\varphi = Pr_{\bowtie\theta_0}(\rho)$ - (nested) CSL formula, a, b - parameters for beta prior, α, β - bound on Type I and Type II errors, assumption constants c, d

Output: Answer if $s \models \varphi$ in $\mathcal{T}$ with confidence α, β

```
1  begin
2      if φ is nested CSL formula then
3          Compute α_ϱ, β_ϱ ∈ (0,1) such that θ_0 − α_ϱ, θ_0 + β_ϱ ∈ (0,1)
4          ᾱ, β̄ = Comp_Top_Error(ϱ, α_ϱ, β_ϱ)
5          for ψ in Top(φ) and state s' of T do
6              t_{ψ,s'} = Bayes_SMC_Nested(s', T, ψ, a, b, α_ψ, β_ψ, c, d)
7          Set n = 0, m = 0
8          Set L_ϱ to be the depth of (non-nested part of) ϱ
9          while True do
10             Increment n
11             Generate a (random) path σ from state s in T of length L_ϱ
12             if σ satisfies ρ using {t_{ψ,s'}} then
13                 Increment m
14             Set B_min = BayesFactorMin(θ_0 + β_ϱ, n, m, a, b, α_ϱ, β_ρ, c, d)
15             Set B_max = BayesFactorMax(θ_0 − α_ϱ, n, m, a, b, α_ϱ, β_ρ, c, d)
16             Set r_1, r_2 = ProbBounds(θ_0, a, b, α_ϱ, β_ρ)
17             if B_min > 1/(r_1 β) then
18                 return true
19             if B_max < α/r_2 then
20                 return false
21     else
22         return Bayes_SMC_Base(s, T, φ, a, b, α, β)
```

We can summarize the correctness criterion of Algorithm 2 as follows:

Theorem 4. *Let s, $\mathcal{T}$, φ, a, b, α, and β, be as in Algorithm 2. If $s \models \varphi$ in $\mathcal{T}$, then*

Algorithm 2 outputs true with probability at least $(1-\alpha)$. *If* $s \not\models \varphi$ *in* $\mathcal{T}$, *then Algorithm 2 outputs false with probability at least* $(1-\frac{1}{\beta})$.

Chapter 7

Experimental Analysis

In this section, we discuss the experimental analysis of the Bayesian approach for statistical model-checking of DTMC models with respect to CSL formulae.

7.1 Case study

We mainly focus on comparing our Bayesian SMC algorithm with the Frequentist SMC (sequential probability ratio test with acceptance sampling[3]) and a probability model checker (PRISM[40]). The case study we consider for our experiments is a robot-navigation system in a grid environment as in Figure 3.1. We generalize the example to an $n \times n$ grid, where we fix diagonally opposite cells as the initial position r of the robot and the destination/goal g, respectively. We randomly assign the label b some of the $n \times n$ cells. We aim to verify the following specification

$$\varphi = Pr_{\geq \theta_0}[Pr_{\geq \theta_1}\ [true\ U^{\leq k_1}\ b]\ U^{\leq k_2}\ g]$$

The formula specifies the property that the robot reaches the destination region labeled with g within k_2 steps with a probability of at least θ_0 while maintaining at least a θ_1 probability of periodically reaching the state labeled with b within steps smaller than k_1.

7.2 Implementation

We have implemented the Bayesian SMC Algorithms 1 and 2 (for non-nested and nested cases) in Python including implementations of random trace generation for the DTMC, checking whether a trace satisfies an LTL formula, and the Bayes Factor Computation. Our experiments are conducted with Ubuntu 12.04 OS, Intel R©Pentium(R) CPU B960 2.20GHz× 2 Processor, 2GB RAM.

7.3 Evaluation

We perform experimental comparisons of our Bayesian SMC algorithm with two other algorithms:

- Frequentist SMC[3]: A statistical model-checking approach based on sampling paths of DTMC that uses the Sequential Wald Probability Ratio Test (SPRT)[45] to accept or reject the hypothesis.
- Probabilistic Model-Checking (PMC)[40]: It is an exact method that determines if $s \models \varphi$ using a state-space exploration based algorithm. PRISM is a state-of-the-art tool that performs PMC on DTMCs.

PMC has the advantage that it gives correct answers with probability 1, however, they are computationally expensive due to the exhaustive state-space exploration. SMC Algorithms are typically light weight computationally, but can only provide correct answers with certain confidence. Next, we compare the performance of these algorithms by varying different parameters. First, we compare our Bayesian approach for different priors.

7.3.1 Evaluation of the priors

We vary the grid size ($n \times n$) and compare the performance of uniform, left-skewed, right-skewed and bell-shaped priors in terms of their inference, number of samples and time as reported in Table 7.1. On all grid sizes, experiments with all of the priors leads to the

same inference (in fact, it agrees with the results returned by PMC). The times are reported in seconds. As expected, the number of samples and time for inference (until one of the conditions on B is satisfied) increases as the grid size/DTMC size increases. Between the priors, we observe that with uniform priors we are able to obtain the inference with the fewest samples and smallest times.

Table 7.1: *Comparison between Bayesian SMC with different priors*

	Uniform prior		Left-skewed prior		Right-skewed prior		Bell-shaped Prior		Status
n	Samples	Time	Samples	Time	Samples	Time	Samples	Time	
2	24	0.475	58	1.005	62	1.35	38	0.69	T
4	4115	95.80	3423	71.00	3623	73.25	3015	63.80	F
8	6376	156.07	12972	309.70	13212	315.75	10195	223.16	F
16	20204	459.11	33348	780.41	34102	791.11	26512	540.05	F

Uniform prior: $a = b = 1$, Left-skewed: $a = 5, b = 2$, Right-skewed: $a = 2, b = 5$, Bell-Shaped prior: $a = b = 2$, $\theta_0 = 0.10$, $\theta_1 = 0.35$, $k_1 = 3$, $k_2 = 8$, error bounds $\alpha = \beta = 0.01$

7.3.2 Varying grid sizes, thresholds and time bounds

Next, we compare the performance of our Bayesian SMC approach with the Frequentist SMC method and PMC in PRISM. For our proposed Bayesian SMC and frequentist SMC, we fix both Type-I and Type-II error bounds to be 0.01. For our Bayesian SMC, we use uniform prior ($a = 1$, $b = 1$ for Beta distribution shape parameters), since, from our previous observations that gives the best performance. For Frequentist SMC, we specify the indifference region parameter, δ, to be 0.01 and 0.001. Our main reason for comparison of our Bayesian SMC with the PMC method is to see if both methods return the same result. PRISM is a mature tool that implements a PMC algorithm, hence, we expect its implementation to be more efficient than our naive implementation, and therefore, the running times might be slightly skewed toward PMC.

We compare the performance of the three approaches on varying grid sizes n, probability thresholds θ_0, θ_1 and time bounds k_1, k_2; the number of samples to arrive at a decision is reported for each of the SMC approaches and the time to arrive at a decision is reported

for all the three approaches. These are tabulated in Tables 7.2, 7.3 and 7.4, for varying grid sizes, probability thresholds and times bounds, respectively. All the methods infer the same correct answers (correct, since, they agree with PMC answers), hence, they are reported only once in the status column.

From Table 7.2, we observe that as the grid size increases, that is, as the size of the DTMCs increases, the number of samples increases for both the SMC based approaches. However, Bayesian SMC requires fewer samples to arrive at the conclusion than Frequentist SMC. In terms of verification time, Bayesian SMC does better for large systems, which indicates that the approach is more scalable.

From Table 7.3, we note that the threshold values have little bearing on the verification time. There appears to be no particular relation between the threshold values and the number of samples to arrive at the answer. Again, as observed from Table 7.4, the depth do not affect the number of samples or the times for SPRT based SMC and PMC, however, the number of samples and time required to arrive at the inference reduces with increasing depths (time bounds) for Bayesian SMC, which again points to its scalability with respect to larger time bounds in the formulae.

Table 7.2: *Comparison between Bayesian SMC, SPRT SMC and PMC for different grid sizes*

	Bayesian SMC		SPRT with $\delta = 0.01$		SPRT with $\delta = 0.001$		PMC	
n	Samples	Time	Samples	Time	Samples	Time	Time	Status
2	23	0.43	167	12.69	1655	12.005	10.85	T
4	2196	49.10	2892	41.15	37247	43.71	37.19	F
8	8097	170.51	19090	168.24	161618	169.20	150.07	F
16	14755	379.79	57052	711.12	642068	765.00	598.88	F

$a = 1$, $b = 1$, $\theta_0 = 0.10$, $\theta_1 = 0.35$, $k_1 = 3$, $k_2 = 8$, Errors $\alpha = \beta = 0.01$

Table 7.3: *Comparison between Bayesian SMC, SPRT SMC and PMC for different probability thresholds*

	Bayesian SMC		SPRT with $\delta = 0.01$		SPRT with $\delta = 0.001$		PMC	
(θ_0, θ_1)	Samples	Time	Samples	Time	Sample	Time	Time	Status
$(0.10, 0.35)$	2923	60.72	5226	41.49	40936	41.59	37.07	F
$(0.10, 0.55)$	3275	65.59	3507	41.57	90084	41.59	38.09	F
$(0.10, 0.75)$	244	4.12	2191	41.47	31060	41.49	37.09	F
$(0.25, 0.45)$	2132	39.76	2457	41.46	23856	41.47	37.09	F
$(0.75, 0.45)$	2398	50.00	2450	41.38	23423	41.30	38.05	F

$n = 4$, $k_1 = 3$, $k_2 = 8$, error bounds $\alpha = \beta = 0.01$

Table 7.4: *Comparison between Bayesian SMC, SPRT SMC and PMC for different time bounds*

	Bayesian SMC		SPRT with $\delta = 0.01$		SPRT with $\delta = 0.001$		PMC	
(k_1, k_2)	Samples	Time	Samples	Time	Sample	Time	Time	Status
$(3, 8)$	1727	35.56	3451	41.21	37677	41.41	36.99	F
$(4, 10)$	1544	37.19	2578	41.51	22660	41.57	39.75	F
$(5, 12)$	993	19.49	1912	41.54	17592	41.58	37.31	F
$(6, 14)$	439	7.66	1106	41.55	11768	42.35	37.20	F

$n = 4$, $\theta_0 = 0.10$, $\theta_1 = 0.35$, error bounds $\alpha = \beta = 0.01$

Chapter 8

Conclusions

The main contribution of the report is a Bayesian statistical model-checking algorithm for verifying properties of Discrete-time Markov Chains specified in Continuous Stochastic Logic. The previous approaches focused on frequentist SMC for the full logic or Bayesian approach for the non-nested fragment of the logic. Our approach extends the previous results to the case of nested probabilistic operators and Bayesian approach. Our preliminary experimental results confirm the benefits of the Bayesian approach as compared to a frequentist approach as well as the traditional probabilistic model checking based on state-space exploration in terms of both the sample complexity and verification time.

There are several interesting future directions, which we briefly discuss below:

- Implement the Bayesian approach for the full logic with multiple levels of nested operators, and perform elaborate experimental comparison involving more complex formulas.
- Extend the approach to more complex models for autonomous systems such as Markov Decision Processes and Stochastic Hybrid Systems that capture non-determinism, stochasticity and dynamics.
- Extend the approach to CSL* which can be defined analogous to CTL*. CTL* extends CTL by relaxing the requirement that the temporal operator be applied only on state formulae to allowing them to be applied on path formulae as well.

- Extend the approach to more complex logics that specify properties of multi-agent systems such as HyperCSL. Hyper-properties specify multi-trace properties as compared to linear-time properties that specify single trace properties. Investigating Bayesian SMC algorithms for probabilistic hyper-properties is an interesting future direction.

Bibliography

[1] Dorsa Sadigh, Katherine Driggs-Campbell, Alberto Puggelli, Wenchao Li, Victor Shia, Ruzena Bajcsy, Alberto L. Sangiovanni-Vincentelli, S. Shankar Sastry, and Sanjit A. Seshia. Data-driven probabilistic modeling and verification of human driver behavior. In *Formal Verification and Modeling in Human-Machine Systems, AAAI Spring Symposium*, March 2014.

[2] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: A tool for automatic verification of probabilistic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2006.

[3] Håkan LS Younes and Reid G Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *International Conference on Computer Aided Verification*, 2002.

[4] Sumit K Jha, Edmund M Clarke, Christopher J Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A bayesian approach to model checking biological systems. In *International Conference on Computational Methods in Systems Biology*, 2009.

[5] Paolo Zuliani, André Platzer, and Edmund M Clarke. Bayesian statistical model checking with application to simulink/stateflow verification. In *International conference on Hybrid systems: computation and control*, 2010.

[6] Gul Agha and Karl Palmskog. A survey of statistical model checking. *ACM Transactions on Modeling and Computer Simulation*, 2018.

[7] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In *International conference on runtime verification*, 2010.

[8] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *International Conference on Computer Aided Verification*, 2004.

[9] Håkan LS Younes and Reid G Simmons. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation*, 2006.

[10] Alexandre David, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, Jonas Van Vliet, and Zheng Wang. Statistical model checking for networks of priced timed automata. In *International Conference on Formal Modeling and Analysis of Timed Systems*, 2011.

[11] David Henriques, Joao G Martins, Paolo Zuliani, André Platzer, and Edmund M Clarke. Statistical model checking for markov decision processes. In *Quantitative Evaluation of Systems*, 2012.

[12] Alexandre David, Dehui Du, Kim G Larsen, Axel Legay, Marius Mikučionis, Danny Bøgsted Poulsen, and Sean Sedwards. Statistical model checking for stochastic hybrid systems. *arXiv preprint arXiv:1208.3856*, 2012.

[13] Moshe Y Vardi. Automatic verification of probabilistic concurrent finite state programs. In *Symposium on Foundations of Computer Science*, 1985.

[14] Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM (JACM)*, 1995.

[15] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999. ISBN 0-262-03270-8.

[16] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.

[17] Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:102–111, 1994.

[18] Holger Hermanns, Joost-Pieter Katoen, Joachim Meyer-Kayser, and Markus Siegle. A markov chain model checker. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pages 347–362, 2000.

[19] Christel Baier and Marta Z. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.

[20] Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Probabilistic analysis of large finite state machines. In *DAC*, pages 270–275, 1994.

[21] Gary D. Hachtel, Enrico Macii, Abelardo Pardo, and Fabio Somenzi. Symbolic algorithms to calculate steady-state probabilities of a finite state machine. In *EDAC - The European Conference on Design Automation, ETC - European Test Conference, EUROASIC - The European Event in ASIC Design, Proceedings, February 28 - March 3, 1994, Paris, France*, pages 214–218, 1994.

[22] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with PRISM: a hybrid approach. *STTT*, 6(2):128–142, 2004.

[23] Christel Baier, Edmund M. Clarke, Vasiliki Hartonas-Garmhausen, Marta Z. Kwiatkowska, and Mark Ryan. Symbolic model checking for probabilistic processes. In *Automata, Languages and Programming, 24th International Colloquium, ICALP'97, Bologna, Italy, 7-11 July 1997, Proceedings*, pages 430–440, 1997.

[24] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Process Algebra and Probabilistic Methods, Performance Modeling and Verification: Joint International Workshop*, 2001.

[25] Yossi Aviv and Awi Federgruen. The value iteration method for countable state markov decision processes. *Operations Research Letters*, 1999.

[26] Diego Bello and German Riano. Linear programming solvers for markov decision processes. In *Systems and Information Engineering Design Symposium*, 2006.

[27] Yasin Abbasi-Yadkori, Peter L Bartlett, and Alan Malek. Linear programming for large-scale markov decision problems. In *JMLR Workshop and Conference*, 2014.

[28] Andrea Bianco and Luca de Alfaro. Model checking of probabalistic and nondeterministic systems. In *Foundations of Software Technology and Theoretical Computer Science, 15th Conference, Bangalore, India, December 18-20, 1995, Proceedings*, pages 499–513, 1995.

[29] Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time markov chains. *ACM Trans. Comput. Logic*, 1(1):162–170, July 2000. ISSN 1529-3785.

[30] Christel Baier, Boudewijn Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, June 2003.

[31] Christian Dehnert, Daniel Gebler, Michele Volpato, and David N. Jansen. On abstraction of probabilistic systems. In *Stochastic Model Checking. Rigorous Dependability Analysis Using Model Checking Techniques for Stochastic Systems - International Autumn School, ROCKS 2012, Vahrn, Italy, October 22-26, 2012, Advanced Lectures*, pages 87–116, 2012.

[32] Mark Kattenbelt, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. A game-based abstraction-refinement framework for markov decision processes. *Formal Methods in System Design*, 36(3):246–280, 2010.

[33] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. Reachability analysis of probabilistic systems by successive refinements. In *Proceedings of the Joint International Workshop on Process Algebra and Probabilistic Methods, Performance Modeling and Verification*, PAPM-PROBMIV '01, pages 39–56, 2001. ISBN 3-540-42556-X.

[34] Christel Baier, Joost-Pieter Katoen, and Holger Hermanns. Approximate symbolic model checking of continuous-time markov chains. In *Proceedings of the 10th International Conference on Concurrency Theory*, CONCUR '99, pages 146–161. Springer-Verlag, 1999.

[35] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Quantitative analysis with the probabilistic model checker PRISM. *Electr. Notes Theor. Comput. Sci.*, 153(2): 5–31, 2006.

[36] Joost-Pieter Katoen, Ivan S Zapreev, Ernst Moritz Hahn, Holger Hermanns, and David N Jansen. The ins and outs of the probabilistic model checker mrmc. *Performance evaluation*, 2011.

[37] J-P Katoen, Maneesh Khattri, and I S Zapreev. A markov reward model checker. In *Quantitative Evaluation of Systems*, 2005.

[38] David Parker. Verification of probabilistic real-time systems. *Real-time Systems Summer School*, 2013.

[39] Marta Kwiatkowska, Gethin Norman, and David Parker. Probabilistic symbolic model checking with prism: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 2004.

[40] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International conference on computer aided verification*, 2011.

[41] Holger Hermanns, Joost-Pieter Katoen, Joachim Meyer-Kayser, and Markus Siegle. Etmcc: Model checking performability properties of markov chains. In *null*, 2003.

[42] Håkan LS Younes. Ymer: A statistical model checker. In *International Conference on Computer Aided Verification*, 2005.

[43] David N Jansen, Joost-Pieter Katoen, Marcel Oldenkamp, Mariëlle Stoelinga, and Ivan Zapreev. How fast and fat is your probabilistic model checker? an experimental performance comparison. In *Haifa verification conference*, 2007.

[44] Ernst Moritz Hahn, Holger Hermanns, Björn Wachter, and Lijun Zhang. Pass: Abstraction refinement for infinite probabilistic models. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, 2010.

[45] Abraham Wald. Sequential tests of statistical hypotheses. *The annals of mathematical statistics*, 1945.

[46] Koushik Sen, Mahesh Viswanathan, and Gul Agha. Vesta: A statistical model-checker and analyzer for probabilistic systems. In *International Conference on the Quantitative Evaluation of Systems*, 2005.

[47] Paolo Ballarini, Benoît Barbot, Marie Duflot, Serge Haddad, and Nihal Pekergin. Hasl: A new approach for performance evaluation and model checking from concepts to experimentation. *Performance Evaluation*, 2015.

[48] Marta Kwiatkowska, Gethin Norman, and David Parker. Stochastic model checking. In *International School on Formal Methods for the Design of Computer, Communication and Software Systems*, 2007.