STATISTICAL APPROACHES FOR PROBABILISTIC MODEL CHECKING

Vincent NIMAL
Worcester College
vincent.nimal@comlab.ox.ac.uk

Oxford University
Computing Laboratory

SUBMITTED IN PARTIAL FULFILMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

September 2010

Abstract

In this dissertation, we survey the different techniques used to perform statistical probabilistic model checking, prove their respective validity and compare them in terms of (statistical) reliability. We then propose some extensions of these methods for computation of rewards, which to our knowledge has not been done before. All these strategies are added to the code of the probabilistic model checking tool PRISM, and we experiment them on several large biological model examples. We conclude by some recommendations for the practical use of these methods, based on the experiments observation and the methods comparisons.

Key words:

statistical model checking, approximative model checking, probabilistic model checking, confidence interval, sequential probability ratio test, Wald's test, rewards, Hoeffding bound, truncation method, Bartlett's test

Acknowledgements

I would like to thank Dr David Parker for the supervision of this thesis. He proposed me an interesting and challenging subject, introduced the PRISM code to me, proofread this dissertation and was always present to advise me in the direction to give to my researches.

During this project, I studied a part of the method that APMC implements, and I really appreciated the explanations of Dr Sylvain Peyronnet about this and the choices he and his team made when they wrote this software.

I am grateful to Mr Bernard Sufrin for the supervision of my Master in general and for his advice and discussions all along the year.

I would also like to thank my French engineering school, the École Nationale Supérieure pour l'Industrie et l'Entreprise (ENSIIE), for offering me the opportunity to follow this Master of Science in the University of Oxford.

Finally, many thanks to my family, my friends and all the people in the Computing Laboratory for their smiles, their kindness and their support.

Contents

List of Figures

List of Equations

Chapter 1

Introduction

Probabilistic model checking is a quantitative automatic formal verification technique, which analyses the validity of properties over mathematical models of stochastic systems. These systems can be randomized communication protocols, like Bluetooth or FireWire, some consensus algorithms or biological mechanisms. Most of the probabilistic model checking algorithms are polynomial, *e.g.* linear equations systems resolution.

It is easy to think that, once we have translated a problem - such as a linear equations system - into a linear problem with matrices, we only have to compute numerically the matrices resolution to get the results. Unfortunately, even some polynomial algorithms can take centuries to end - especially if we are talking about $10^{10} \times 10^{10}$ (*a priori* non sparse) matrices.

This is the kind of problem modellers can face when they try to model check some complicated models with millions of states on a numerical probabilistic model checker. Several ways to deal with this: the model can be abstracted, leading to fewer states but possibly representing a too much generalized problem. Another possibility is to use the power of statistics to approach at lower cost the probabilities to compute.

It is indeed possible to apply Monte Carlo method to this context, after having clearly defined the random space (here, the paths in the model). We do not have to work on the full model and heavily test properties on it, we just pick some paths, test them linearly and compute an average.

Having an estimation of the result is useful, but how far can we trust this value? Was the estimator converging after the $N = 10542^{nd}$ loop, or can we assume it is reliable to accept it at the 34^{th} one? These questions call for a discussion about the estimation and its reliability in statistical model checking.

In this dissertation, we propose to answer this reliability question by confidence intervals and statistical tests. These statistics objects provide their own parameters of reliability, and can be computed in the context of probabilistic model checking. Several methods can be found in the literature, and we will prove, implement and compare them practically on real-size examples issued from biology.

Probabilities offer an interesting information about a property over a model, but we can sometimes want to get a different result - like the time the system takes to do something, the number of states it goes across... The cost/reward structure can be adapted to this problem. But it works in a different way, and all the previous methods are not compatible with it. This dissertation proposes some extensions to these methods to compute statistically rewards.

In the next chapter, the probabilistic model checking basics will be introduced to get an overview of how this mechanism works in classic problems, and which problems we are going to consider here. These problems can be classified into 4 groups:

- the estimation of a probability $P_{=?}[\phi]$ (chapter 3), which answers questions like *"What is probability that this state is reached within 12 time units?"*;
- the estimation of a reward $R_{=?}[\phi]$ (chapter 4), which solves problems like *"How many times must I expect to toss a coin to simulate a 6-sided die?"*;
- the test of property with bounded probability $P_{\bowtie\theta}[\phi]$ (chapter 5), which claims whether a property like *"Every run goes through that state with a probability lower than 0.3"* is true or not;
- the test of property with bounded reward $R_{\bowtie\theta}[\phi]$ (chapter 6), which states the validity of a property like *"The expected number of servers to repair within the first 10 hours is lower than 3"*.

The dissertation concludes with a table which summarizes the different problems, the methods to use (or not to use) in terms of the conditions and their observed efficiency.

Chapter 2

Background

Model checking can be presented as a technique which can extract from models some unexpected behaviour. This method is in particular used to find bugs in very complicated and critical software. The technique requires

- a *model* $\mathcal{M}$, generally a graph $\mathcal{G}$ of possible states $\mathcal{V}$, linked by transitions defined by $\delta \subseteq \mathcal{V} \times \mathcal{V}$;
- some *logic expressions* with atoms in A labelling certain states of the graph: $L : \mathcal{V} \to 2^A$;
- a *property* $\mathcal{P}$, expressed in a temporal logic;
- a *model checker*, which will check whether the property holds or not on this labelled model: $\mathcal{M}^L \models \mathcal{P}$.

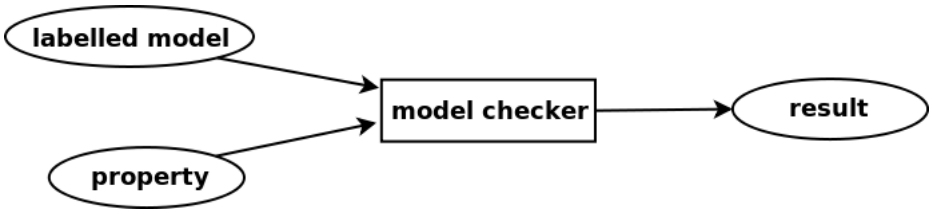


Figure 2.1: General model checking process

Figure 2.1 shows graphically the model checking process. For a comprehensive introduction to this general method, we can read for example [1].

In this study, we focus on *probabilistic model checking*. This is a particular instance of model checking, in which models can represent some stochastic behaviour. As presented in section 2.1 of this chapter, we add probability measure to the model and get a Markov chain. This is specifically convenient for random communication protocols (for example, Bluetooth [6]), randomized consensus [24] or bioinformatics problems expression [14, 5, 26].

The labels we add to the model follow a classic boolean logic, and the atoms represent some specific states or events linked to the problem the Markov chain modelled.

The temporal logic has to be adapted to the probability measure. Generally, we use extensions of classical temporal logics CTL[1], LTL[2] and CTL*[3]: PCTL, PLTL, PCTL* and CSL[4] (for a comprehensive presentation of the different grammars, read [1, 17]). The formulae can also be bounded (in terms of transitions in general model checking, or in time, in continuous time Markov chains, for instance), and we can define sub-logics from these. In this study, we restrict the formulae to a subset of bounded PCTL/CSL, which is defined in section 2.2. We will see that it can be extended anyway.

The model checking methods to compute the probability of having $\mathcal{M} \models \mathcal{P}$ need also to be adapted to the probabilistic logics. Section 2.3 describes the headlines of the numerical techniques used for instance in PRISM [17, 23], which is a numerical probabilistic model checker.

The described numerical techniques to compute probabilities are unfortunately not scaling well with the number of states of the Markov chain [33, 11]. An alternative is to approach statistically these measures with Monte Carlo simulations - this is the object of section 2.4.

Note that we only consider the frequentist statistics here. There are also some interesting results using Bayesian approaches, which are out of the scope of this thesis - read for instance [14, 35].

2.1 Model: DTMCs and CTMCs

A labelled model (called generally Kripke structure) in model checking is completely defined by the tuple $\langle \mathcal{V}, \mathcal{V}_{initial}, \delta, L \rangle$. The transition between two states is non deterministic by nature. [17] replaces the non-deterministic transitions by transitions with a probability: $P : \mathcal{V} \times \mathcal{V} \to [0,1]$. The graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{V}_{initial}, P, L \rangle$

[1] Computation Tree Logic.
[2] Linear Temporal Logic.
[3] An upper set of CTL and LTL.
[4] Continuous Stochastic Logic.

becomes a labelled *discrete time Markov chain* (DTMC).

A finite *path* is an execution of the Markov chain. [17] provides a probability measure for these paths: $\forall$ non null finite path $\omega = s_0 \dots s_n$, with the $s_i \in \mathcal{V}$,

$$P(\omega) = \begin{cases} 1 & \text{if } n = 0 \\ P(s_1) \times \dots \times P(s_n) & \text{otherwise.} \end{cases}$$

The probability that a (bounded) property ϕ holds in a model $\langle \mathcal{V}, \mathcal{V}_{initial}, P, L \rangle$, starting from a state in $\mathcal{V}_{initial}$, can be then quantified by the number of paths starting in $\mathcal{V}_{initial}$ and satisfying ϕ. Note that even though this study only requires finite paths in the graph, this can be extended to infinite paths with the notion of cylinders [17].

In the next chapters, we will illustrate the DTMC operations with the *Knuth and Yao* algorithm [15] (abbreviated to *dice.pm*), which simulates a fair six-sided die with a (supposed) fair coin.

DTMCs use discrete transitions relating some states. *Continuous time Markov chains* (CTMCs) suppose there is a continuous time t, and relate each pair of states with exponential distributions. The real parameter of these distributions are given by $R : \mathcal{V} \times \mathcal{V} \to \mathbb{R}^+$. The probability of going from a state x to a state y within t (time units) is $1 - e^{-R(x,y) \times t}$.

A CTMC is then defined by $\langle \mathcal{V}, \mathcal{V}_{initial}, R, L \rangle$. The notion of path is extended to this graph. The probabilistic measure associated to it is the cylinder, and depends on the different states the path go through and the time it stays in each of them. For the formal definition, read [17, 23].

A state is likely to be related to several states - this is a *race condition* [17]. Given the exponential distributions of the different possible incoming states, we can compute the probability to go from one state to another one. This defines the *embedded DTMC* of the CMTC. More formally, its relation P is built with:

$$\forall x \in \mathcal{V}, \forall y \in \mathcal{V}, P(x,y) = \begin{cases} 1 & \text{if } \sum_{s' \in \mathcal{V}} R(s,s') = 0 \text{ and } x = y \\ \dfrac{R(x,y)}{\sum_{s' \in \mathcal{V}} R(s,s')} & \text{if } \sum_{s' \in \mathcal{V}} R(s,s') \neq 0 \\ 0 & \text{otherwise.} \end{cases}$$

This relates the CTMCs to DTMCs.

In the next chapters, model checking on CTMCs is illustrated by biological examples, the same ones that [14] used in its benchmark:

- the *circadian rhythm* (denoted by *circadian.sm* in the following - read [2, 31] for details);
- the *Cell cycle control* (denoted by *cyclin.pm* - see [20] for details);
- the *Fibroblast Growth Factor Signalling Model* (denoted by *fgf.sm* - read [9, 10] for details).

Getting a probability of a formula is useful, but sometimes we need to get an expectation (over the paths) of a parameter of the model - a power consumption, a number of lost packets, a number of steps required to end... The *reward structure* is a pair $\langle \rho, \iota \rangle$, where $\rho \in \mathcal{V} \longmapsto \mathbb{R}^+$ defines the value of this parameter we have in every state of $\mathcal{V}$, and $\iota \in \mathcal{V} \times \mathcal{V} \longmapsto \mathbb{R}^+$ defines the value of the parameter during a transition between two states of the graph. We can compute some expectations with these functions using summations (see [17]).

For the rewards, we use a model of a dependable cluster of workstations (denoted by *cluster.sm*), due to [8].

2.2 Properties: subset of bounded PCTL/CSL

The examples we study allow us to restrict the properties to a very small subset of bounded PCTL for DTMCs and bounded CSL for CTMCs. We will indeed consider the properties recognized by the grammar

$$\begin{aligned} property ::= & P_{=?}[F^{\leq t}\, proposition] \\ & | P_{\bowtie \theta}[F^{\leq t}\, proposition] \\ proposition ::= & label | proposition \wedge label. \end{aligned}$$

$F^{\leq t}\phi$ on a path ω means ϕ *is eventually true within* t, with t an integer for DTMCs with PCTL or a real for CTMCs with CSL. More formally,

$$\omega \models F^{\leq t}\phi \equiv \exists t_0 \leq t . \omega(t_0) \models \phi,$$

where $\omega(t)$ is the state in ω occupied at time (or step) t.
$P_{=?}[\phi]$ represents the probability that ϕ holds in the model, and $P_{\bowtie\theta}[\phi]$ is the property stating that the probability that ϕ holds in the model is $\bowtie \theta$, with θ a probability and $\bowtie \in \{<, \leq, >, \geq\}$.

Note that it is certainly acceptable to have the temporal operators bounded in the context of simulation, because these models are simulating some behaviours

with approximations and it is not relevant to try to obtain answers above a certain limit of time.

The probability operators are not nested, because the meaning (and the reliability) of having a certain approximated probability of getting an approximated probability is not very clear - and the biological examples we consider do not use this feature.

Table 2.2 describes the respective properties we test over the models.

Model:	Formula:	Meaning:
circadian.sm	$P_{=?}[F^{\leq 0.25} ma > 5]$	"probability that the number of activated messenger RNAs exceeds 5 units within 0.25 time units"
cyclin.pm	$P_{=?}[F^{\leq 0.5} cyclin\_bound > 3]$	"probability that the number of bond Cyclin molecules exceeds 3units within 0.5 time units"
fgf.sm	$P_{=?}[F^{\leq 20} FRS2\_GRB > 0 \wedge relocFRS2 = 0 \wedge degFRS2 = 0]$	"probability that Grb2 binds to FRS2 within 20 time units"

Figure 2.2: Table of the properties - and their meanings - tested over the models

There are two types of rewards[5] we compute in our examples:

- the *cumulated* one until t ($R_{=?}[C^{\leq t}]$), which is the expectation of the structure after t time units;
- the *cumulated* one until ϕ ($R_{=?}[F\phi]$), which is the expectation before a state satisfies ϕ.

The formulae we use are following a subset of PCTL/CSL extended with rewards ([17, 23]):

$$expectation ::= R_{=?}[Flabel] | R_{\bowtie\theta}[Flabel] \\ | R_{=?}[C^{\leq t}] | R_{\bowtie\theta}[C^{\leq t}].$$

Table 2.3 describes the respective formulae we test over the models.

[5]There is also the *immediate* reward at time t, but we do not consider it here.

Model:	Formula:	Reward structure:	Meaning:
dice.pm	$R_{=?}[Fs=7]$	$\rho(x)=0, \forall x \in \mathcal{V}$ $\iota(x,y) = \begin{cases} 1 \text{ if } P(x,y) \neq 0 \\ 0 \text{ otherwise} \end{cases}$	"expected number of steps required to get the die value"
cluster.sm	$R_{num\_repairs=?}[C^{\leq 10}]$	$\rho(x)=0, \forall x \in \mathcal{V}$ $\iota(x,y) = \begin{cases} 1 \quad \text{if } P(x,y) \neq 0 \\ \quad \wedge x.r = true \\ \quad \wedge y.r = false \\ 0 \quad \text{otherwise} \end{cases}$	"expected number of servers to repair within 10 hours"

Figure 2.3: Table of the rewards - and their meanings - computed through the models

2.3 Probabilistic Model Checking: overview

Probabilistic model checking is performed using different graphs and numerical algorithms, depending on the property it has to check. Here, we only consider properties of form $P[F^{\leq t}\phi]$.

For DTMCs, recall that the probability we want to compute is defined by the number of satisfying paths in the model. Let $Sat(\phi)$ denote the set of states from $\mathcal{V}$ which respects ϕ. Computing the probability of having $F^{\leq n}\phi$ is the same as computing the probability of having a path going in $Sat(\phi)$ within the n first steps. This reaching probability can be found by solving the following recursive equation ([23], instance of bounded Until model checking[6]):

$$Prob(x,step) = \begin{cases} 1 & \text{if } x \in Sat(\phi) \\ 0 & \text{if } k < 0 \\ \sum_{y \in \mathcal{V}} P(x,y) \times Prob(y, step-1) & \text{otherwise.} \end{cases}$$

Practically, we create a matrix P' which is a copy of P with all its transitions from a state in $Sat(\phi)$ to another one equal to 1. If ψ is a vector representing all the states from $\mathcal{V}$, with $\psi_i = 1$ if $s_i \in Sat(\phi)$ and 0 otherwise, we compute the probability of reaching $Sat(\phi)$ from each state by n matrices multiplication:

$$Prob(s_i, n) = \left[(P')^k . \psi \right]_i .$$

[6] Indeed, $F^{\leq n}\phi \equiv true U^{\leq n}\phi$.

For CTMCs, the concept is the same, but with continuous values ([23], instance of bounded Until model checking):

$$Prob(x,t) = \begin{cases} 1 & \text{if } x \in Sat(\phi) \\ \displaystyle\int_0^t \underbrace{\left(P^{emb(\mathcal{M})}(x,y) \times E(x) \times e^{-E(x)\times u}\right)}_{\text{probability of moving from } x \text{ to } y \text{ at time } u} \times \underbrace{Prob(y,t-u)}_{\substack{\text{probability in } y \text{ of} \\ \text{satisfying } \phi \text{ before} \\ t-u \text{ tine units elapse}}} du & \text{otherwise,} \end{cases}$$

with $E(x) = \sum_{y \in \mathcal{V}} P(x,y)$.

These integrals can be solved numerically, using for instance trapezoidal or Simpson and Romberg integration [23]. This technique is however expensive, and could present some problems of numerical stability. Another approach, implemented in PRISM, derives first a new CTMC from the studied CTMC, in which all the states in $Sat(\phi)$ are absorbing (*i.e.* a path going to such a state remains in this state forever). Concretely, the rates matrix R is replaced by R', a copy of R in which all the relations starting with a state in $Sat(\phi)$ are 0. We have then:

$$Prob(x_i,t) = \left[\underbrace{\Pi_t^{CTMC}}_{\substack{\text{matrix of} \\ \text{transient probability}}} .\psi \right]_i,$$

where $\Pi_t^{CTMC}(x,y)$ is the probability of being in state y, starting from x, at time t. We compute practically Π_t^{CTMC} via the uniformisation [23]:

$$\Pi_t^{CTMC} = \sum_{i=0}^{\infty} \gamma_{q.t,i} \times \left(P^{unif(CTMC)}\right)^i.$$

The rewards $R[C^{\leq t}]$ and $R[F^{\leq t}\phi]$ are the expectation of the random variables

$$X_{C^{\leq t}}(\omega) = \begin{cases} 0 & \text{if } t = 0 \\ \sum_{i=0}^{t-1} \rho(x_i) + \iota(x_i, x_{i+1}) & \text{otherwise,} \end{cases}$$

and

$$X_{F\phi}(\omega) = \begin{cases} 0 & \text{if } x_0 \in Sat(\phi) \\ \infty & \text{if } x_i \notin Sat(\phi), \forall i \geq 0 \\ \displaystyle\sum_{i=0}^{min\{j | x_j \models \phi\}-1} \rho(x_i) + \iota(x_i, x_{i+1}) & \text{otherwise.} \end{cases}$$

The first one is computed with matrices multiplication:

$$E[X_{C^{\leq t}}] = \begin{cases} 0 & \text{if } t = 0 \\ \rho + (P \circ \iota).1_n + P.E[X_{C^{\leq t-1}}] & \text{if } t \geq 0 \end{cases}$$

where $\circ$ denotes the pointwise matrix multiplication.

The second one can be computed with graph analysis and linear equation system [23].

2.4 Statistical approaches: generation and analysis

Probabilistic model checking offers a very accurate answer (the only approximation is due to the numerical truncation in the computation). But it requires to build the DTMC or CTMC of the model (see figure 2.4 [23]). It is acceptable for models with no more than 10^7 states, but it is (currently) hardly realizable above. Unfortunately, some biological problems, for instance, require models with 10^9 states or above.

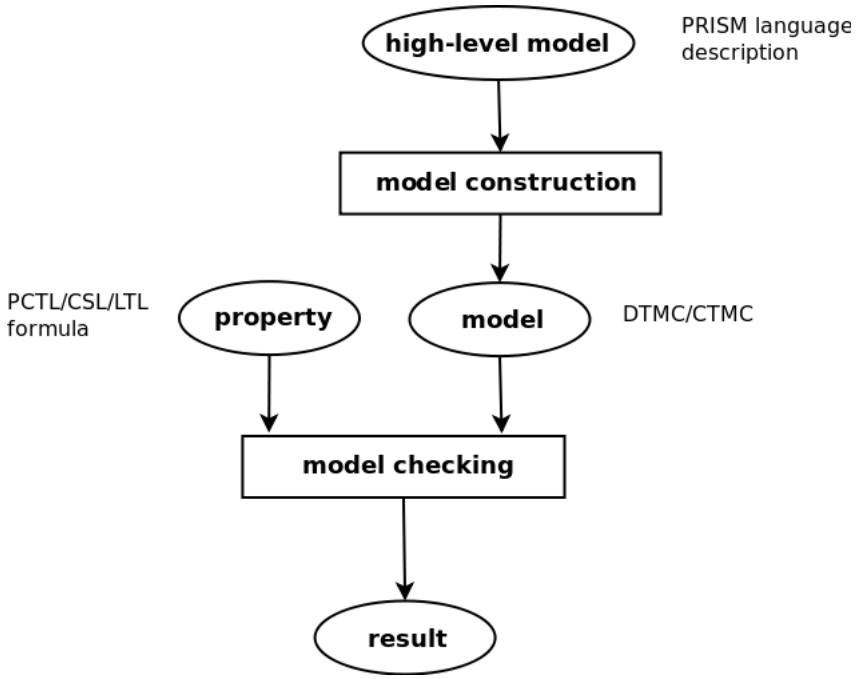


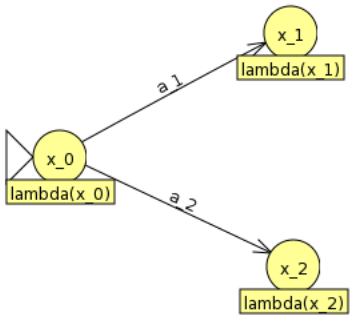
Figure 2.4: Global mechanism of probabilistic model checking

An interesting idea consists in approximating the probability we try to compute with a statistic estimator relying on a limited number of events from the random space. This is the Monte Carlo method. In our case, the limited number of events are runs (or paths) of the model. The major advantage is that we can extract these paths directly from the high-model[7] without building the graph. We name this step *generation*.

For CTMCs[8], the following generation algorithm can be used [5]:

Algorithm 1 Paths generation algorithm for a CTMC

1. init $t = 0$, $x = x_0$
2. generate different h_i such that $\{h_i\}_i$ is a sample of realizations of the random variable $h \rightsquigarrow exp(\lambda(x))$
3. generate the successor state, i.e. m_i such that $\{m_i\}_i$ sample of the random variable Z with $Prob(Z = m_i) = \frac{\alpha_{m_i}(x)}{\lambda(x)}$
4. set $t = t + h$, $x = u_m(x)$ and go to 2.



Once we have extracted the paths, we estimate the probability with testing the property over each of them - it is the *verification* step. For example, we compute $\frac{r}{N}$, where r is the number of paths which validate property and N the total number of extracted paths (the sample size). The paths are linear, so the verification is fast to perform.

Thus, to perform *statistical model checking*, we basically only need to extract these runs and compute the mean.

Now, the estimator we chose converges to the real probability asymptotically. So taking a limited number of runs offers no absolute guarantee about the result. It

[7] Expressed in the PRISM description language, for instance.

[8] Extracting from a DTMC is very similar.

seems necessary to get an idea of the reliability that an experimentation offers for a certain number of runs. This reliability can be characterized for example by

- a confidence interval, with its level of confidence and width;
- an asymptotic confidence interval, similar to the normal one;
- a statistical test, with its two types of errors.

Also, for a given level of reliability, it could be interesting to run the optimal number of steps required. This step is the *analysis* part of the statistical probabilistic model checking.

Figure 2.5 sketches the global mechanism of the statistical probabilistic model checking. This dissertation assumes that generation and verification steps are already realized[9], and focus on the analysis step.

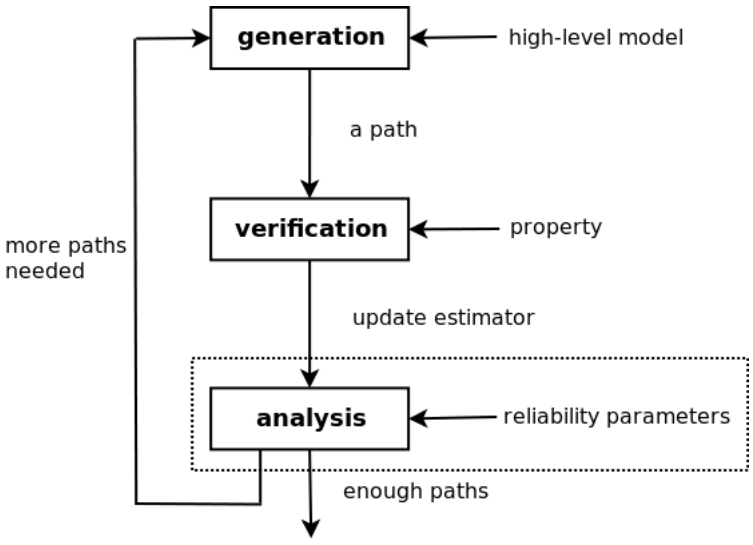


Figure 2.5: Global mechanism of statistical model checking

[9]It was the case in PRISM.

Chapter 3

Statistical Model Checking: $P_{=?}[\phi]$ problem

The first problem we study is the case $P_{=?}[\phi]$: given a formula ϕ, we want to estimate the probability that this formula holds in the model. Having an estimation is useful, but it is dangerous to use it directly. We do not know how close it is to the real probability. To quantify the reliability, confidence intervals will be introduced in section 3.1. Because extracting lots of paths from the model can become expensive, the question of the optimal number of steps required to get a certain confidence interval will be asked in section 3.2. [11] proposes another approach to probability estimation, from which we can derive a confidence interval in a different way. Section 3.5 will present this method, and section 3.6 will compare the different approaches, illustrated by some experiments done with PRISM on the biological examples. This chapter ends with the presentation of other related problems.

3.1 Confidence interval methods

A confidence interval (denoted CI in the following) is an estimated interval of a certain width $2w$ such that, if we repeat the estimation several times, the real probability is in this interval $100 \times (1-\alpha)\%$ of the times[1]. α is the level of confidence. Figure 3.1 sketches a representation of intervals. This interval, associated to an estimated value (here the means), represents the uncertainty of this value.

[1] It is often improperly referred as "the real probability has a probability $1-\alpha$ of lying in this interval".

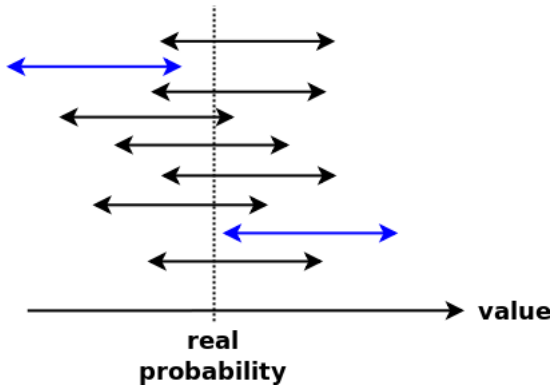


Figure 3.1: Some estimated CIs, containing the real probability. The blue CIs are among the $100 \times \alpha\%$ which do not contain it.

3.1.1 Confidence interval (CI)

Our objective in this section is to compute an interval for some given α, which represents the degree of confidence we can have in the result, and w, which symbolises the accuracy of the estimation.

Suppose we have extracted a path ω from the model. We test the property ϕ over ω. Let

$$\chi_p(\omega) = \begin{cases} 1, & \text{if } \omega \models \phi \\ 0, & \text{otherwise} \end{cases}$$

be a Bernouilli random variable. We are looking for the expectation of this variable, $E[\chi_p]$. Let $\{Y_i\}_i$ be a set of realizations of the random variables $X_i := \chi_p$ defined by

$$\forall i \in \{1,\dots,N\}, Y_i = \chi_p(\omega_i).$$

We assume that the Y_i are independent and identically distributed (IID) and normally distributed[2]. Thus, $\overline{Y} = \frac{1}{N}\sum_{i=1}^{N} Y_i$ is an unbiased and consistent estimator. So $E[\overline{Y}] = E[\chi_p]$, and, if $\sigma^2 = Var[\chi_p]$, Central-limit theorem gives

$$Z = \frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{\sigma^2}{N}}} \rightsquigarrow \mathcal{N}(0,1).$$

For a confidence level $\alpha \in [0;1]$, $1-\alpha = Prob(|Z| \leq z) = Prob\left(|\overline{Y} - E[\chi_p]| \leq z\sqrt{\frac{\sigma^2}{N}}\right)$.

[2]This relies on the generator of pseudo random paths.

The confidence interval for a level α is then

$$CI = \left[\overline{Y} - z_{1-\alpha/2} \sqrt{\frac{\sigma^2}{N}}; \overline{Y} + z_{1-\alpha/2} \sqrt{\frac{\sigma^2}{N}} \right], \tag{3.1}$$

where $z_{1-\alpha/2}$ is a quantile from Normal distribution.

Actually, we do not know σ^2, so we estimate it with

$$S^2 = \frac{1}{N-1} \sum_{i=1}^{N} (Y_i - \overline{Y})^2.$$

The random variable becomes

$$Z' = \frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{S^2}{N}}}$$

and follows a Student distribution with $N-1$ degrees of freedom: $Z' \rightsquigarrow T_{N-1}$. The confidence interval for a level α is then

$$CI = \left[\overline{Y} - t_{N-1,1-\alpha/2} \sqrt{\frac{S^2}{N}}; \overline{Y} + t_{N-1,1-\alpha/2} \sqrt{\frac{S^2}{N}} \right], \tag{3.2}$$

where $t_{N-1,1-\alpha/2}$ is a quantile from Student distribution with $N-1$ degrees of freedom. This is one of the intervals we implement in our PRISM: if we are given α and w, we iterate until

$$|CI| = \left| 2t_{N-1,1-\alpha/2} \sqrt{\frac{S^2}{N}} \right| \leq 2w,$$

so the bound of the CI method is

$$\frac{t^2_{N-1,1-\alpha/2} S^2}{w^2} \leq N. \tag{3.3}$$

We have then $\overline{Y}$, and $CI = [\overline{Y} \pm w]$.

3.1.2 Asymptotic confidence interval (ACI)

In [5], it was deduced from $S^2 \to \sigma^2$ when $n \to +\infty$ that

$$Z = \frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{S^2}{N}}} \rightsquigarrow_{n\to+\infty} \mathcal{N}(0,1)$$

approximately[3], and $Prob(|Z| \leq z) \approx 1-\alpha$, so the quantile $z_{1-\alpha/2}$ from Normal distribution is used instead of the Student one. [26] implemented a technique using the same formula. The bound becomes then:

$$\frac{t^2_{N-1,1-\alpha/2} S^2}{w^2} \leq N. \tag{3.4}$$

This is actually the *asymptotic confidence interval*[4], that we also implement in PRISM. [19], which proposes the same approximation p. 233, notes however that there is some perceptible differences between CIs and ACIs practically, CIs being more exact. [7], chapter 5, presents it as a misuse of Normal distribution.

In [25], in the context of Bernouilli variables, it is proposed to substitute the estimator S^2 of the variance by an estimator built with the empirical mean: $\hat{\sigma}^2 = \overline{Y}(1-\overline{Y})$ (close to the property of Bernouilli variables $Var[X] = E[X](1-E[X])$). This is asymptotically equivalent, because here

$$\hat{\sigma}^2 = \frac{n-1}{n} S^2.$$

Indeed,

$$\begin{aligned} S^2 &= \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \overline{Y})^2 \\ &= \frac{1}{n-1} \left(\sum_{i=1}^{n} Y_i^2 - \frac{1}{n} \left(\sum_{j=1}^{n} Y_j \right)^2 \right) \\ &= \frac{1}{n-1} \left(\sum_{i=1}^{n} Y_i - \frac{1}{n} \left(\sum_{j=1}^{n} Y_j \right)^2 \right) \end{aligned}$$

[3] That is to say for N "large enough".

[4] There are also credible intervals, using Bayesian theory [35], and prediction intervals, roughly the same but with a frequentist approach. These intervals are out of the scope of this dissertation.

because $Y_i^2 = Y_i$, and

$$\hat{\sigma}^2 = \overline{Y}(1-\overline{Y}) = \frac{1}{n}\left(\sum_{i=1}^{n} Y_i - \frac{1}{n}\left(\sum_{j=1}^{n} Y_j\right)^2\right).$$

So, for $N > 1$,

$$\frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{S^2}{N}}} \rightsquigarrow T_{N-1} \Leftrightarrow \frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{\hat{\sigma}^2}{N-1}}} \rightsquigarrow T_{N-1}.$$

3.1.3 Computation optimization

CI and ACI require the estimator S^2, but its computation requires $\overline{Y}$ and all the Y_i up to here - which means we should store them, and that could raise a problem of memory space if there are lots of iterations to perform[5]. To compute efficiently S^2, [5] proposes an optimization. We have

$$S^2 = \frac{\sum_{i-1}^{n} Y_i^2}{n-1} - \frac{\left(\sum_{i-1}^{n} Y_i\right)^2}{n(n-1)}.$$

$Y_i = 0$ or 1, so $Y_i^2 = Y_i$, and if r is the number of traces ω such that $\chi_p(\omega) = 1$, $\overline{Y} = \frac{r}{n}$ and $S^2 = \frac{r}{n-1} - \frac{r^2}{n(n-1)} = \frac{r(n-r)}{n(n-1)}$.

So we just need to store r and n during the iterations to compute S^2 and $\overline{Y}$ on the fly. We increment them through iterations.

3.2 How many steps to get a confidence interval?

We know how to compute the CI. Now, what is the minimum number of steps (*i.e.* paths extracted from the model) required to compute a CI? If we fix the width w of the $CI = [a;b]$ to $\frac{|b-a|}{E[\chi_p]} \leq w$, and for a confidence level α, we have $Prob(|Z'| \leq t_{1-\alpha})$, leading to [5]

$$E[\chi_p] \in CI = \left[\overline{Y} - t_{N-1,1-\alpha/2}\sqrt{\frac{S^2}{N}}; \overline{Y} + t_{N-1,1-\alpha/2}\sqrt{\frac{S^2}{N}}\right] \Rightarrow 2t_{N-1,1-\alpha/2}\sqrt{\frac{S^2}{N}} \leq wE[\chi_p]$$

$$\Leftrightarrow \frac{4t_{N-1,1-\alpha/2}^2 S^2}{E[\chi_p]^2 w^2} \leq N.$$

[5] In the following, we compute up to 10^9 iterations.

Of course, the smaller w is, the more useful the CI is. But it also requires more steps.

We considered here $\frac{|b-a|}{E[\chi_p]}$ and not $|b-a|$ to work on a kind of normalized interval and to fix a CI width w independent from the value of $E[\chi_p]$. Indeed, if $E[\chi_p] \neq 0$, $Prob(E[\chi_p \in CI]) = 1-\alpha \Leftrightarrow Prob(E[1 \in \frac{CI}{E[\chi_p]}]) = 1-\alpha$. However, this raises two problems:

- The first problem is that we obviously ignore $E[\chi_p]$. We can simply consider absolute w, i.e. $w \geq |b-a|$.
- A second one is that S^2 depends on N. This requires to check the inequality at each iteration.

For the first problem, if we know that there is a $\delta \in [0;1]$ such that $E[\chi_p] \geq \delta$, then we have

$$\frac{4t^2_{N-1,1-\alpha/2}S^2}{E[\chi_p]^2 w^2} \leq \frac{4t^2_{N-1,1-\alpha/2}S^2}{\delta^2 w^2}.$$

So if we have a N such that $\frac{4t^2_{N-1,1-\alpha/2}S^2}{\delta^2 w^2} \leq N$, then it implies $\frac{4t^2_{N-1,1-\alpha/2}S^2}{E[\chi_p]^2 w^2} \leq N$.

Instead of "normalizing" following the real value of $E[\chi_p]$, [26] proposes to normalize by the estimated value $\overline{Y}$, closer to the CI: we get

$$\frac{4t^2_{N-1,1-\alpha/2}S^2}{\overline{Y}^2 w^2} \leq N.$$

Then $\frac{S^2}{\overline{Y}^2}$ estimates c_Y^2, the squared coefficient of variation[6] of Y, *i.e.* $\frac{\sigma^2}{E[\chi_p]^2} = \frac{1-E[\chi_p]}{E[\chi_p]}$.

Practically, we preferred to implement with an absolute width in PRISM:

$$\frac{4t^2_{N-1,1-\alpha/2}S^2}{w^2} \leq N.$$

This is more convenient for the graphic representation of what we compute and the comparison between the different tests[7].

Whatever is the bound we use, we always have at least S^2 which depends on N. This is not a problem if we compute the traces one by one and test after each of them. But if we want to parallelize, we will have to fix an arbitrary number m (the number of threads/process) such that the bound is checked all the m "iterations".

[6] With the previous notation, we have $c_Y^2 = \frac{n(n-r)}{r(n-1)}$.

[7] But implementing the relative width using estimated means would have also been valid.

3.3 Another approach: Approximate Model Checking (AMC)

[11] proposes to solve another problem:

$$Prob(|A-\gamma| \leq \varepsilon) \geq 1-\delta,$$

where A is the approximation of the probability computed by their algorithm, γ is the real probability, ε a parameter of approximation and δ a parameter of confidence (we will denote this method AMC).

The solution is based on the Chernoff-Hoeffding bound. [11] uses the formula

$$Prob[|A-\gamma| > \varepsilon] < 2e^{-\frac{N\varepsilon^2}{4}},$$

whereas [12, 22, 18] provide

$$Prob[|A-\gamma| > \varepsilon] < 2e^{-2N\varepsilon^2}.$$

Because the second one imposes fewer iterations practically[8], and [12] proves it, we consider this inequality. We take δ such that $1-\delta$ equals the inequality right-hand side, which leads to a lower bound for the number of samples N required for given parameters ε and δ:

$$N \geq \frac{ln(\frac{2}{\delta})}{2\varepsilon^2}. \tag{3.5}$$

It is easy to see that this is actually computing a confidence interval with a confidence level lower (but unknown) than δ:

$$Prob(\gamma \in [A-\varepsilon; A+\varepsilon]) \geq 1-\delta.$$

The table 3.2 presents the translation between the two problems.

AMC	CI
approximation parameter: ε confidence parameter: δ ($-\delta'$ unknown) sample size: N	half CI width: $\frac{width}{2} = w$ confidence level: α sample size: N

Figure 3.2: Table summing up the AMC and CI correspondences

δ, the level of confidence, is a probability and then must belong to the $[0,1]$ segment. To impose this when not specifying δ explicitly, we have to be sure that we

[8] Indeed, we require N iterations with [11] such that $N \geq 4\frac{ln(\frac{2}{\delta})}{\varepsilon^2}$, whereas [12] requires $N \geq \frac{ln(\frac{2}{\delta})}{2\varepsilon^2}$.

have enough samples for the required approximation: we must have N and ε such that

$$N\varepsilon^2 \geq \frac{ln2}{2}.$$

Indeed, $N = \frac{ln(\frac{2}{\delta})}{2\varepsilon^2} \Rightarrow N\varepsilon^2 = \frac{ln(\frac{2}{\delta})}{2}$, so $0 \leq \delta \leq 1 \Leftrightarrow N\varepsilon^2 \geq \frac{ln2}{2}$.
We do not have this constraint with CI/ACI methods, because $\alpha = 2\left(1 - cdf\left(w\sqrt{\frac{N}{S^2}}\right)\right)$, with the cumulative distributive function $cdf(x) = \frac{1}{2}+$ positive area (see figure 3.3), because $w\sqrt{\frac{N}{S^2}} \geq 0$.

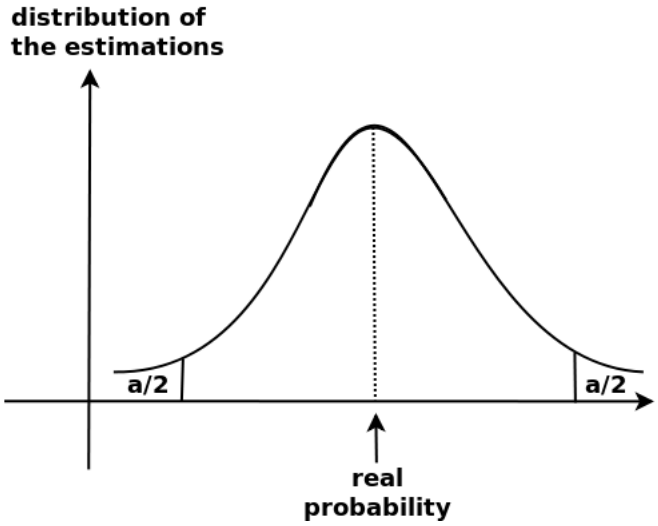


Figure 3.3: Distribution of the estimations.

So $\frac{1}{2} \leq cdf(w\sqrt{\frac{N}{S^2}}) \leq 1$, and $0 \leq \alpha \leq 1$.

The AMC method was already implemented in PRISM. We add to this method the CI and ACI methods, using a statistical library[9], with the bound computed using the mean ($\frac{4t_\beta^2 S^2}{w^2}$).

[9]We were initially using JSC [4], but we now use Colt [36], which has an open source license and is more efficient in our context.

3.4 Decisions and display

Until now, we were only considering $Z' = \frac{\overline{Y} - E[\chi_p]}{\sqrt{\frac{S^2}{n}}} \rightsquigarrow T_{n-1}$, for $Y_1, \ldots, Y_n$ realizations of IID random variables $X_1, \ldots, X_n$. This formula is obviously no more valid if $\sigma^2 = 0$, which means $S^2 = 0$, or $Y_1 = \ldots = Y_n$. This can however happen: if we test a simple model with a property that always or never holds, we have $\chi(\omega) = 0$ for all the paths or 1 for all the paths, and we get $S^2 = 0$. A first idea could be then to decide that if the m first Y_is are equal, then $\overline{Y} = Y_1$. The problem is that we can have a sample starting with m 0s and continuing with 0 or 1, i.e. "non-tautological or antilogical property".

To decide whether we are in $\overline{Y} = Y_1$ case or in a $Z' \rightsquigarrow T_{n-1}$ case, given our confidence intervals (or approximation parameter) width ε, we choose (arbitrarily) to say that if we arrive at a moment with a n such that the next result will not change the mean with respect to ε approximation, then we are in the $\overline{Y} = Y_1$ case. In other words,

$$\left| 1 - \frac{n + Y_{n+1}}{n+1} \right| \leq \varepsilon, \tag{3.6}$$

with $Y_{n+1} = 0$ or 1 (for $\overline{Y} = 1$ case).
So the decision is

$$\begin{aligned} \frac{1 - Y_{n+1}}{n+1} \leq \varepsilon &\Leftrightarrow 0 \leq \varepsilon \wedge \frac{1}{n+1} \leq \varepsilon \\ &\Leftrightarrow n \geq \frac{1}{\varepsilon} - 1. \end{aligned}$$

And we get the same condition for $\overline{Y} = 0$ case ($\left| 0 - \frac{0 + Y_{n+1}}{n+1} \right| \leq \varepsilon$).
Unfortunately, it is still possible (but very rare) to have

$$\underbrace{Y_1 = 0, \ldots, Y_m = 0}_{m}, \underbrace{Y_{m+1} = 1, \ldots, Y_n = 1}_{n-m}$$

such that the 1s are significant for the mean $\overline{Y}$. This automatic decision maker is implemented, but the user has always the possibility to specify manually its decision, which can be useful for limit cases.

Monte Carlo simulations can take time, and it is always good to have an index of the progress through time. With AMC, we can display the progress of the computation because we know before the simulation the number of samples to test. This is not the case in CI and ACI cases. We propose anyway a possible displayer:

since we have $n < t^2_{n-1,1-\alpha/2}\frac{S^2}{\varepsilon^2}$, and the simulation converges to $n \geq t^2_{n-1,1-\alpha/2}\frac{S^2}{\varepsilon^2}$, we can use

$$100 \times \frac{n\varepsilon^2}{t^2_{n-1,1-\alpha/2}S^2} \quad (\leq 100).$$

Obviously, this is not linear, and generally, the closer the simulation is to satisfy the $n \geq t^2_{n-1,1-\alpha/2}\frac{S^2}{\varepsilon^2}$ property, the more numerous the samples are needed to improve the accuracy.

3.5 Experiments and comparisons

In this section, the different methods are compared and tested through the implementation in PRISM. We first illustrate these analyses with a simple model, *poll.sm*, which models a cyclic server polling system [13], and compute the probability that the first station is awaiting service within one time unit, namely $P_{=?}[F^{\leq 1} s1 = 1]$. Then we confirm the observations with the three biological models presented in table 2.2.

3.5.1 How to read the graphs?

If one run of PRISM is enough to get an estimation $\overline{Y}$ of the probability and check the width of the CI $2w$ (here given in parameters), it is not enough to check the level of confidence α. That is why we perform for each test 100 runs of PRISM, and we display the repartition of the computed estimations $\overline{Y}_i$ (see figure 3.4 for example). The bars represent the number of estimations which belong to a subclass[10] of values, the vertical blue dashed line (*theoretical mean*, in the legend) represents the real probability, and the red one (*experimental mean*, in the legend) the average of the estimations[11]. The normal curve is computed by R[12] with these data, and the bold part represents the $1-\alpha$ part of the area - that is to say the estimations $\overline{Y}_i$ such that the real probability is in the corresponding CI, *i.e.* $E[\chi_p] \in CI_i = [\overline{Y}_i \pm w]$. Because all the $\overline{Y}_i$ of the bold area are such that $\overline{Y}_i - w \leq E[\chi_p] \leq \overline{Y}_i + w$, all these estimations must respect $E[\chi_p] - w \leq \overline{Y}_i \leq E[\chi_p] + w$, and the bold part must be contained in the interval centred in the real probability with a width $2w$ (see graphic representation in figure 3.4).

[10] The number of classes used on these graphs was computed automatically by R - but it does not induce anything in our study.

[11] This average is not useful if we know the real probability, as in the first examples. The real value is computed numerically with PRISM. But for the biological models, the numerical approach is out of range, so we consider this average.

[12] R is a free software environment for statistical computing and graphics.

3.5.2 AMC vs. CI

Figure 3.4 shows the distribution of means obtained for 100 runs of PRISM with AMC method. It is clear that the experimental level of confidence δ", materialized by the bold part of the curve, is much lower than the expected one α - which would take almost the whole interval centred in the real probability with a width $2w$. This means there was some useless sampling ($N = 18445$ samples).

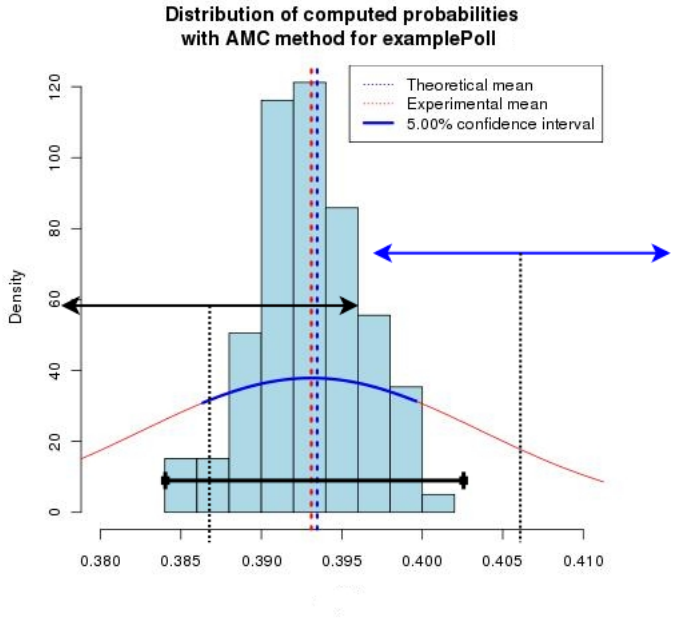


Figure 3.4: Distribution of means with AMC method for $\alpha = 5\%$, $\varepsilon = 10^{-2}$ (which induces exactly $N = 18455$ samples for each run of PRISM), 100 runs of textscPrism for *poll.sm*. All the estimations in the black segment have $E[\chi_p]$ in their CI, *e.g.* the black CI. The red estimation does not have $E[\chi_p]$ in its CI.

We run 100 times PRISM with the CI method, always with example *poll.sm*, the CSL formula $P_{=?}[F^{\leq 1} s_1 = 1]$, and the parameter $\varepsilon = 10^{-2}$ and the confidence level[13] $\alpha = 5\%$. Figure 3.5 presents the distribution (the number of samples, inconstant, was under 9200). As expected, the bold part of the curve is closer to the interval of width $2w$ centred on the real probability.

[13]The slight approximation, α" ≈ α, is due to the fact that we compute until the next integer n such that $n \geq t^2_{n-1,1-\alpha/2} \frac{S^2}{\varepsilon^2}$.

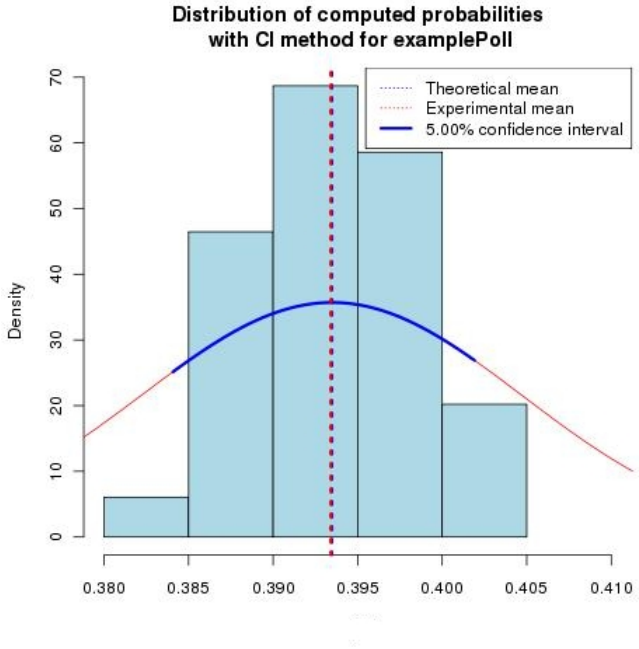


Figure 3.5: Distribution of means with CI method for $\alpha = 5\%$, $\varepsilon = 10^{-2}$ (which induces less than $N = 9200$ samples for each run of PRISM), 100 runs of PRISM for *poll.sm*. The computed α" $\approx \alpha$.

This would mean that the Chernoff-Hoeffding inequality would be too large here. Let us compare the AMC and CI bounds:

AMC	CI
$N \geq \frac{ln(\frac{2}{\alpha})}{2\varepsilon^2}$	$N \geq t^2_{N-1,1-\alpha/2}\frac{S^2}{\varepsilon^2}$
$= 18445$ samples	≤ 9200 samples

We run a Scilab simulation which computes the difference of samples respectively required by one method and the other one:

$$Z = argmin_n \left\{ t^2_{n-1,1-\alpha/2}\frac{S^2}{\varepsilon^2} \leq n \right\} - \frac{ln(\frac{2}{\alpha})}{2\varepsilon^2}.$$

For $\alpha = 5\%$, on the segment $[0.01 : 1]^2$, we notice in figure 3.6 that for small ε^2, we can have an extreme gain of 10^5 samples[14].

[14]Note that the effect of S^2 is not very important here.

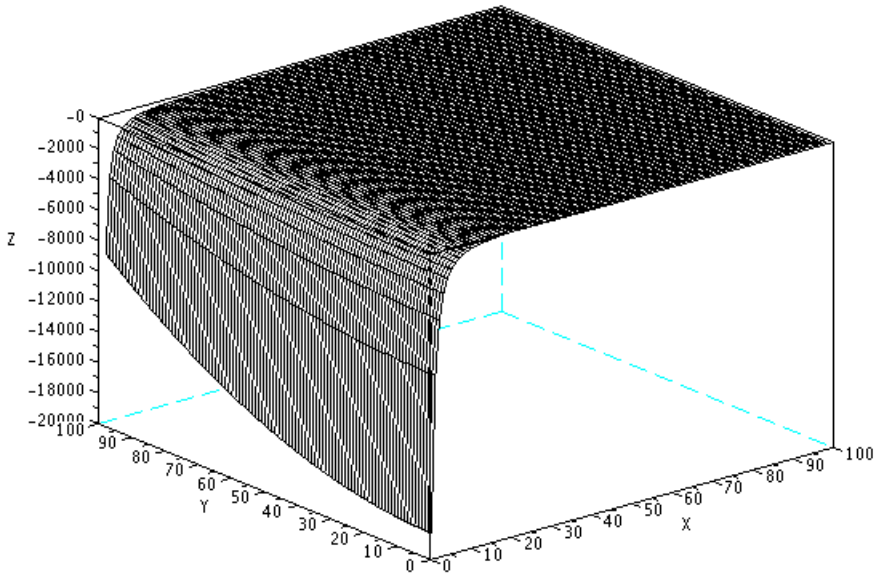


Figure 3.6: Comparison of the AMC and CI bound (Scilab simulation). $X = \varepsilon^2$, $Y = S^2$, $Z = argmin_n \left\{ t^2_{n-1,1-\alpha/2} \frac{S^2}{\varepsilon^2} \leq n \right\} - \frac{ln(\frac{2}{\alpha})}{2\varepsilon^2}$, with $(X,Y) \in [0.01 : 0.5]^2$

If we compute the ratio rather than the difference, we get, in figure 3.7,

$$Z = argmin_n \left\{ t^2_{n-1,1-\alpha/2} \frac{S^2}{\varepsilon^2} \leq n \right\} / \frac{ln(\frac{2}{\alpha})}{2\varepsilon^2}.$$

We maximize the relative gain with S^2 the biggest and ε^2 the smallest.

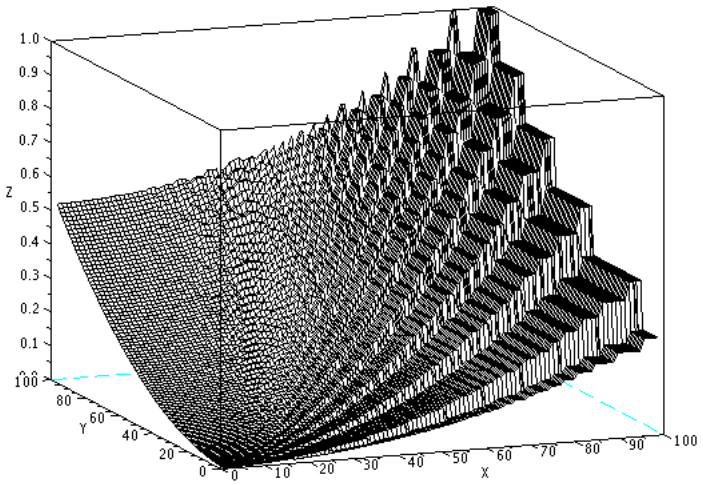


Figure 3.7: Comparison of the AMC and CI bound (Scilab simulation). $X = \varepsilon^2$, $Y = S^2$, $Z = argmin_n \left\{ t^2_{n-1,1-\alpha/2} \frac{S^2}{\varepsilon^2} \leq n \right\} / \frac{ln(\frac{2}{\alpha})}{2\varepsilon^2}$, with $(X,Y) \in [0.01 : 0.5]^2$

So, most of the time, the AMC method computes for far much more iterations than it is required to have, as in CI.

3.5.3 CI vs. ACI

We decide to compute ACI, *i.e.* CI using $\mathcal{N}$ instead of T_{N-1}.

$$ACI = \left[\overline{Y} - z_{1-\alpha/2} \sqrt{\frac{S^2}{N}}; \overline{Y} + z_{1-\alpha/2} \sqrt{\frac{S^2}{N}} \right]. \tag{3.7}$$

This induces the bound (3.4), namely

$$\frac{q^2_{1-\alpha/2} S^2}{w^2} \leq N,$$

and because of Student and Normal distributions[15], $z_{1-\alpha/2} \leq t_{N-1,1-\alpha/2}$ ([19], p. 234), so the number of steps required is likely to be lower.

[15] Actually, $t_{N-1,1-\alpha/2} \to z_{1-\alpha/2}$ when $n \to +\infty$.

Practically, for the same example as previously and with the same parameters, we get the figure 3.8.

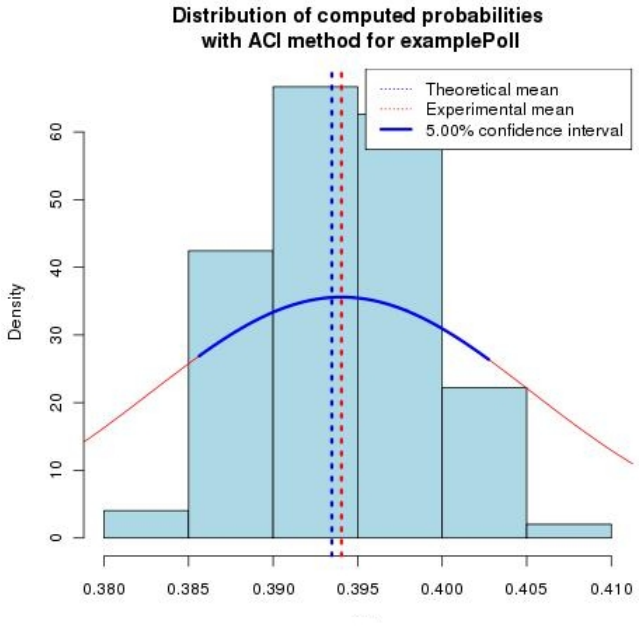


Figure 3.8: Distribution of means with ACI method for $\alpha = 5\%$, $\varepsilon = 10^{-2}$ (which induces less than $N = 9200$ samples for each run of PRISM), 100 runs of PRISM for *poll.sm*. The computed α" $\approx \alpha$.

In figure 3.8, the number of samples required is a little lower than for the CI (see comparison in figure 3.9), but if the confidence level looks unchanged, we notice that the mean is further from the real probability than it was with Student distribution. This is simply due to the fact that Z' does follow $\mathcal{N}(0,1)$ only for large number of samples (these errors can also be seen in the examples of [19]). However, this is not a problem here, because we compute confidence interval - we are not looking for a perfect estimation $\overline{Y}$, just an interval in which we are likely to have the real probability we are looking for $100 \times (1-\alpha)\%$ of the time.

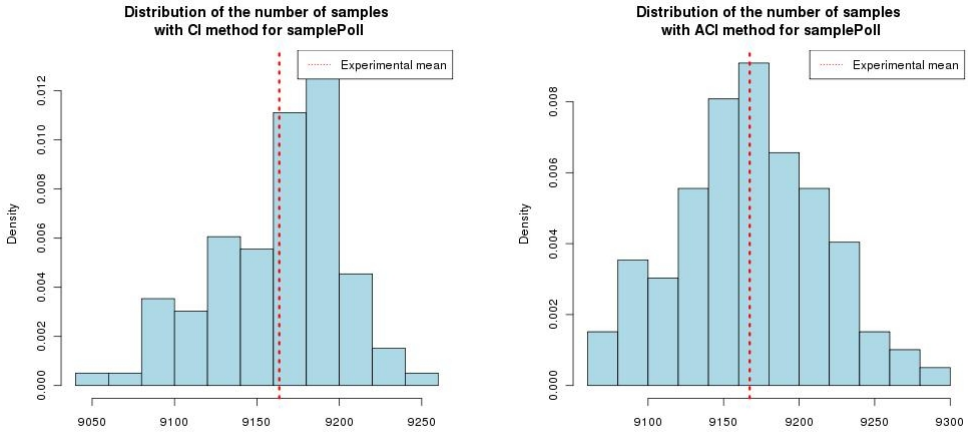


Figure 3.9: Distribution of the number of samples required for CI method (on the left) and ACI method (on the right) for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM for *poll.sm*.

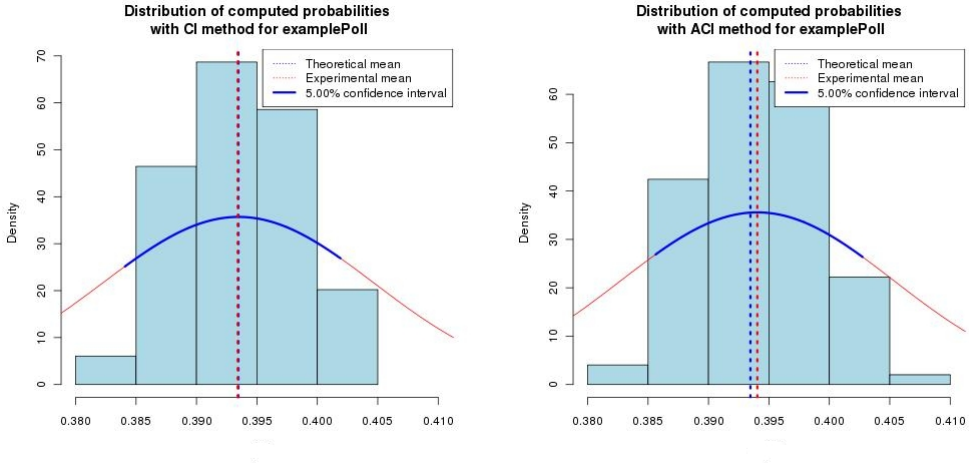


Figure 3.10: Distribution of means with CI method on the left, ACI method on the right for $\alpha = 5\%$, $\varepsilon = 10^{-2}$ (which induces around $N = 9200$ samples for each run of PRISM), 1000 runs of PRISM for *poll.sm*.

In figure 3.10, we make the same experiment, but with 1000 runs. The difference of means is no more perceptible. But we notice that, like in the previous experiment, the confidence interval is larger than the one with CI. This is due to the fact that the confidence interval really computed is not the one we think.

Let us note CI^* the real confidence interval we are looking for. The CI confidence interval computed with CI method is, because of the integer truncation, a bit more accurate than the real one - so, if we centre the estimator[16], $CI^* \supseteq CI$. We have also, with centring, $CI^* \supseteq AMC$. This is convenient, because if the real probability is in CI or AMC, then it is in the CI^* we think we have computed.

This property is unfortunately not verified for ACI: it is possible to have, with centring, $ACI \supseteq CI^*$, as in this example. That is a possible explanation of why it is a little more hazardous to interpret ACI method results instead of CI or AMC methods results.

3.5.4 Tests and comparisons

We extend the experimentations to the biological examples with

- the distribution of computed estimation of probability for 100 runs in figure 3.12;
- the time and number of samples required in figure 3.11;
- the distribution of number of samples required for 100 runs in figure 3.13.

	poll.sm	*circadian.sm*	*cyclin.sm*	*fgf.sm*
AMC	18445 samples / 35 sec	18445 samples / 14 sec	18445 samples / 4 sec	18445 samples / 8 sec
CI	9170 samples / 6 sec	2700 samples / 2 sec	8625 samples / 1 sec	9400 samples / 4 sec
ACI	9160 samples / 7 sec	2600 samples / 2 sec	8625 samples / 1 sec	9350 samples / 4 sec

Figure 3.11: Comparisons of times for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM (test configuration: Core2duo @ 2.1Ghz, 4GB ram, Linux Fedora 10).

[16] Or, without centring the confidence intervals, we can simply say that the widths $w_{CI^*} \geq w_{CI}$.

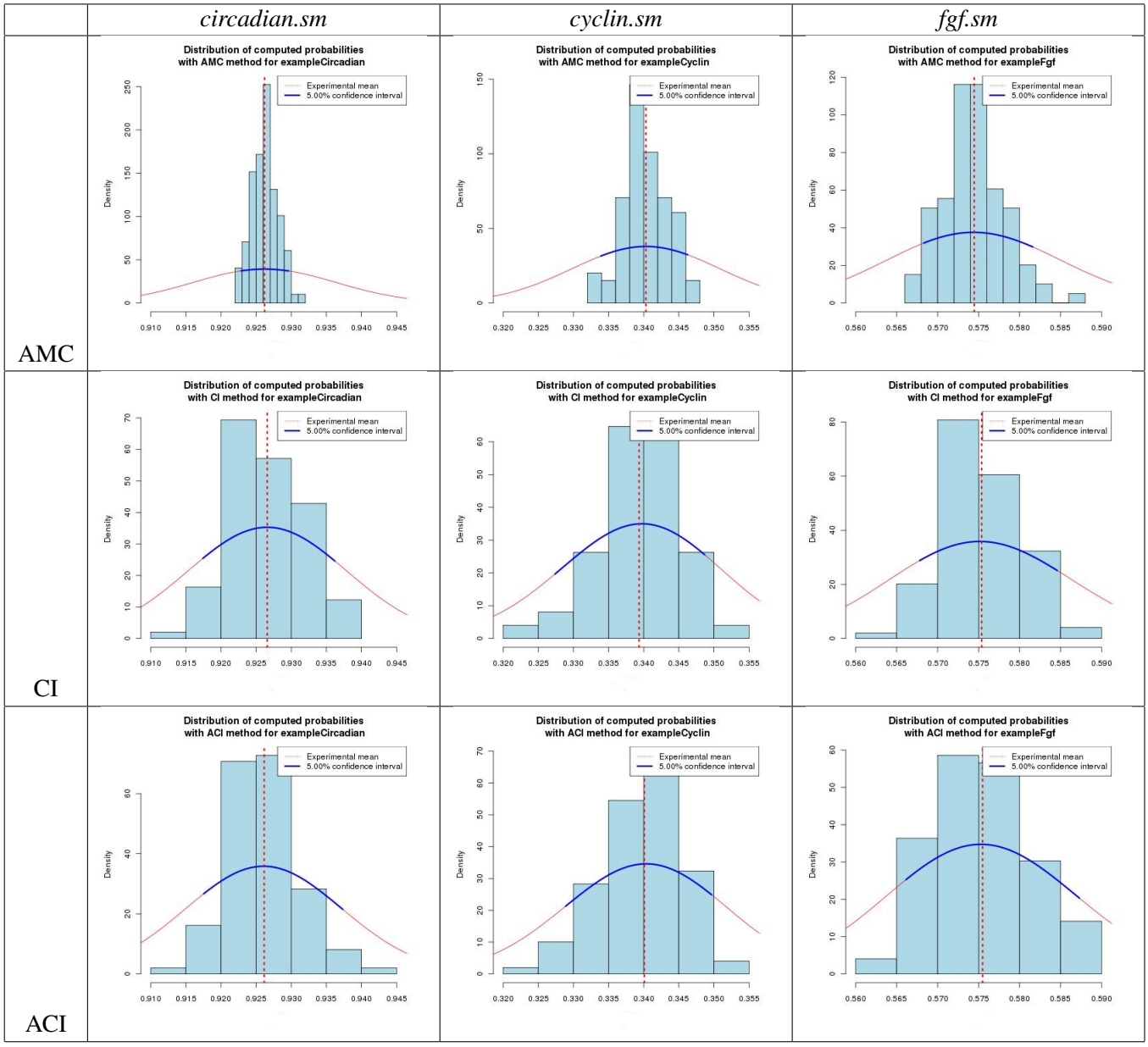


Figure 3.12: Comparisons of distributions for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM.

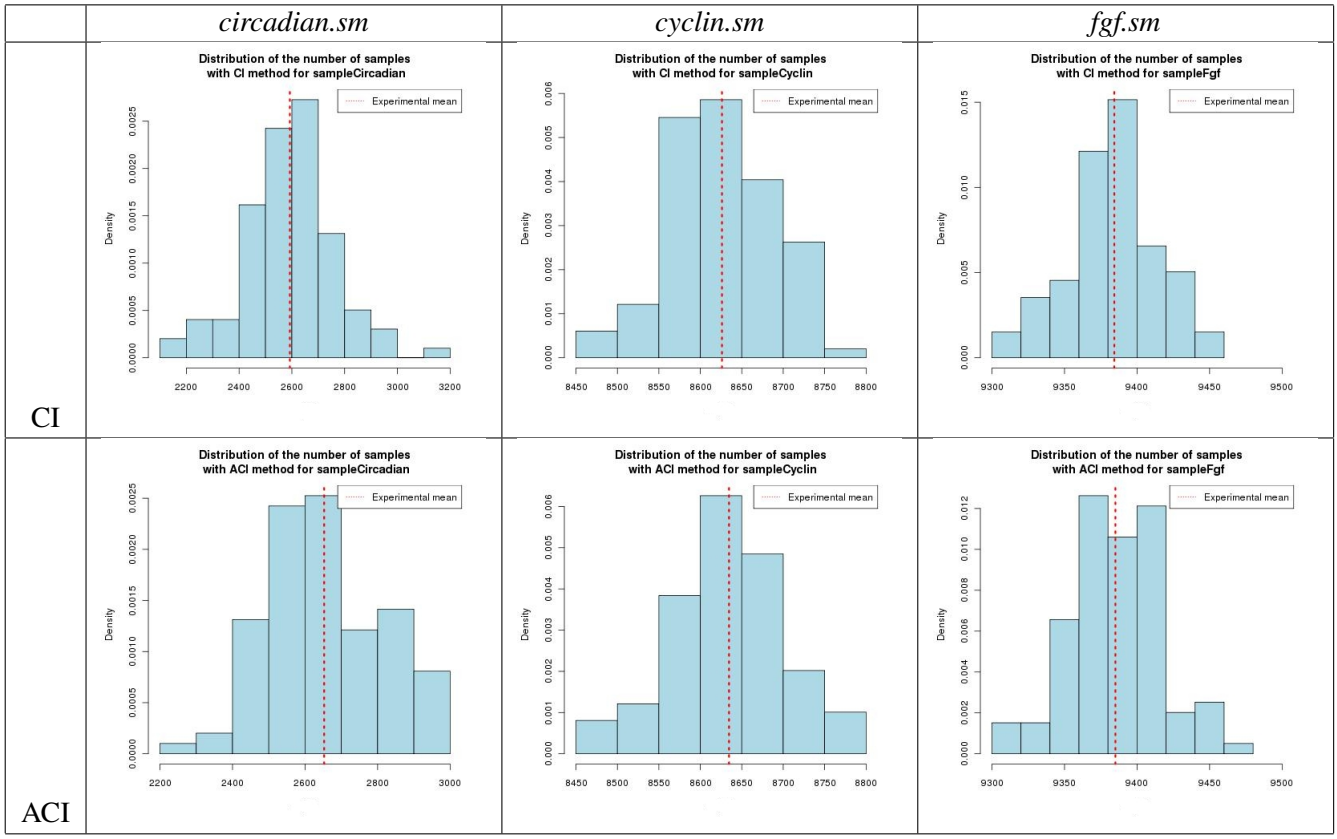


Figure 3.13: Comparisons of distributions of number of samples required for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM.

3.5.5 Analysis

- AMC knows from the beginning the number of loops it must perform. It does not need to do expensive comparison in every loop - it just increments variables. But it performs more loops.
- CI performs fewer loops and is almost optimal in this sense. Unfortunately, it must compare at every loop (to check whether it has reached the bound or not) and must recompute $t_{N-1,1-\alpha/2}$ because the number of degrees of freedom changes - at an expensive cost.
- ACI computes a little fewer loops than CI and, even if it compares at every loop, only compute once the $z_{1-\alpha/2}$ quantile. This is much faster - but unfortunately the confidence interval we really compute is bigger than the one we

ask. It is not convenient, because we can have then the real probability $E[\chi_p]$ in the confidence interval we have computed, but not in the real confidence interval we thought to compute.

We can also find a compromise by only testing during one iteration out of M with CI and ACI. The greater M is, the faster the iterations run, but the more numerous they potentially are.

3.6 Other related problems

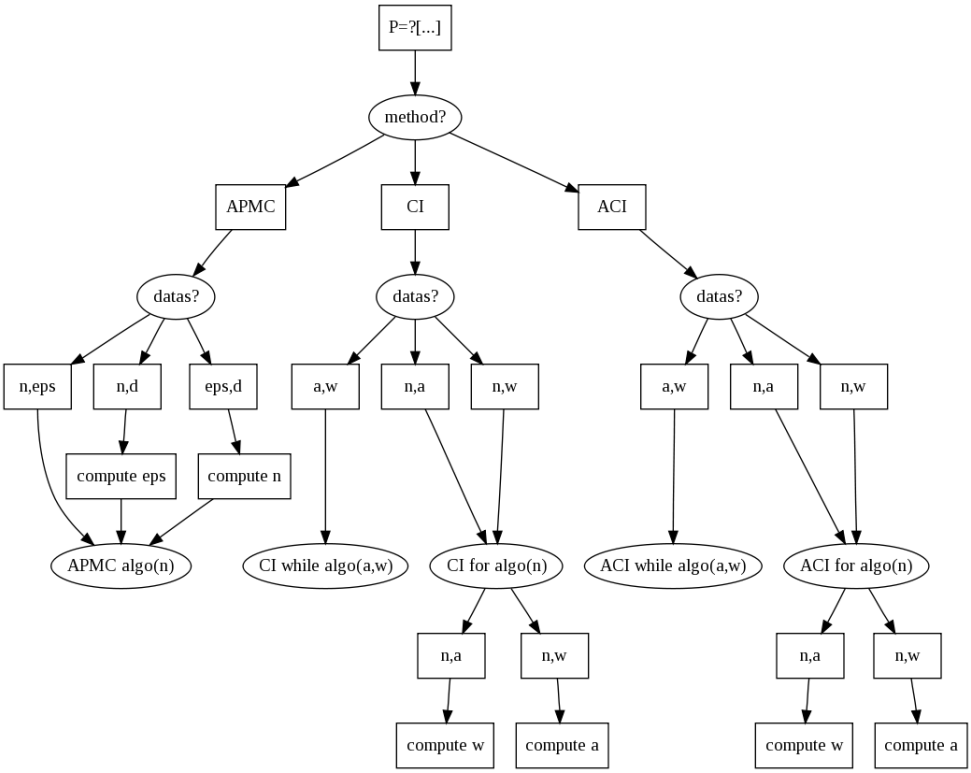


Figure 3.14: Different algorithms to use in terms of given data.

CIs and ACIs computation is also implemented for the case in which the algorithm is given the number of samples and another parameter - either the confidence level or the CI width. As for AMC method, the number of iterations is known before

execution - but we cannot compute the third parameter before iterations as in AMC because we need the S^2 estimator.

The graph 3.14 presents the different possibilities and the order the parameters are computed.

3.6.1 How to compute the missing parameters?

From the bounds of AMC (3.5), CI (3.3) and ACI (3.4), we can derive equations which relate a parameter to two other ones. This allows us to compute the missing parameter.

For AMC:

$$\varepsilon = \sqrt{\frac{ln(\frac{2}{\delta})}{2N}} \tag{3.8}$$

$$\delta = 2e^{-2N\varepsilon^2} \tag{3.9}$$

For CI[17]:

$$w = t_{N-1,1-\alpha/2}\sqrt{\frac{S^2}{N}} \tag{3.10}$$

$$\alpha = 2\left(1 - cdft_{N-1}\left(w\sqrt{\frac{N}{S^2}}\right)\right) \tag{3.11}$$

For ACI[18]:

$$w = z_{1-\alpha/2}\sqrt{\frac{S^2}{N}} \tag{3.12}$$

$$\alpha = 2\left(1 - cdfn\left(w\sqrt{\frac{N}{S^2}}\right)\right) \tag{3.13}$$

[17] *cdft* is the cumulative distributive function of the Student law with $N-1$ degrees of freedom.
[18] *cdfn* is the cumulative distributive function of the Normal law.

Indeed, $t_{N-1,1-\alpha/2} = w\sqrt{\frac{N}{S^2}}$ and $Prob(-t_{N-1,1-\alpha/2} \leq Z' \leq t_{N-1,1-\alpha/2}) = 1-\alpha$, so, using the symmetry of Student distribution,

$$2Prob(Z' \geq t_{N-1,1-\alpha/2}) = \alpha,$$

and

$$Prob(Z' \leq t_{N-1,1-\alpha/2}) = cdft_{N-1}(t_{N-1,1-\alpha/2}) = \frac{1-\alpha}{2},$$

leading to

$$\alpha = 2\left(1 - cdft_{N-1}\left(w\sqrt{\frac{N}{S^2}}\right)\right).$$

And the same for ACI with the normal quantiles.

3.6.2 Implementation

We finally create an interface which summarize all the confidence interval computation methods (figure 3.15), and implement them. All the probability (and reward) computation methods can be used with the same scheme (algorithm 2).

Algorithm 2 General computation algorithm for statistical model checking

```
compute missing parameter before simulation (if AMC)
while the bound of the method is not reached do
    pick a path from the high-level model
    check it
    update the estimators
    print the progress, if relevant
end while
compute missing parameter after simulation (if CI/ACI)
if the property has its probability/reward bounded then
    compare threshold probability/reward and computed mean estimator
    return validity
else
    return mean estimator
end if
```

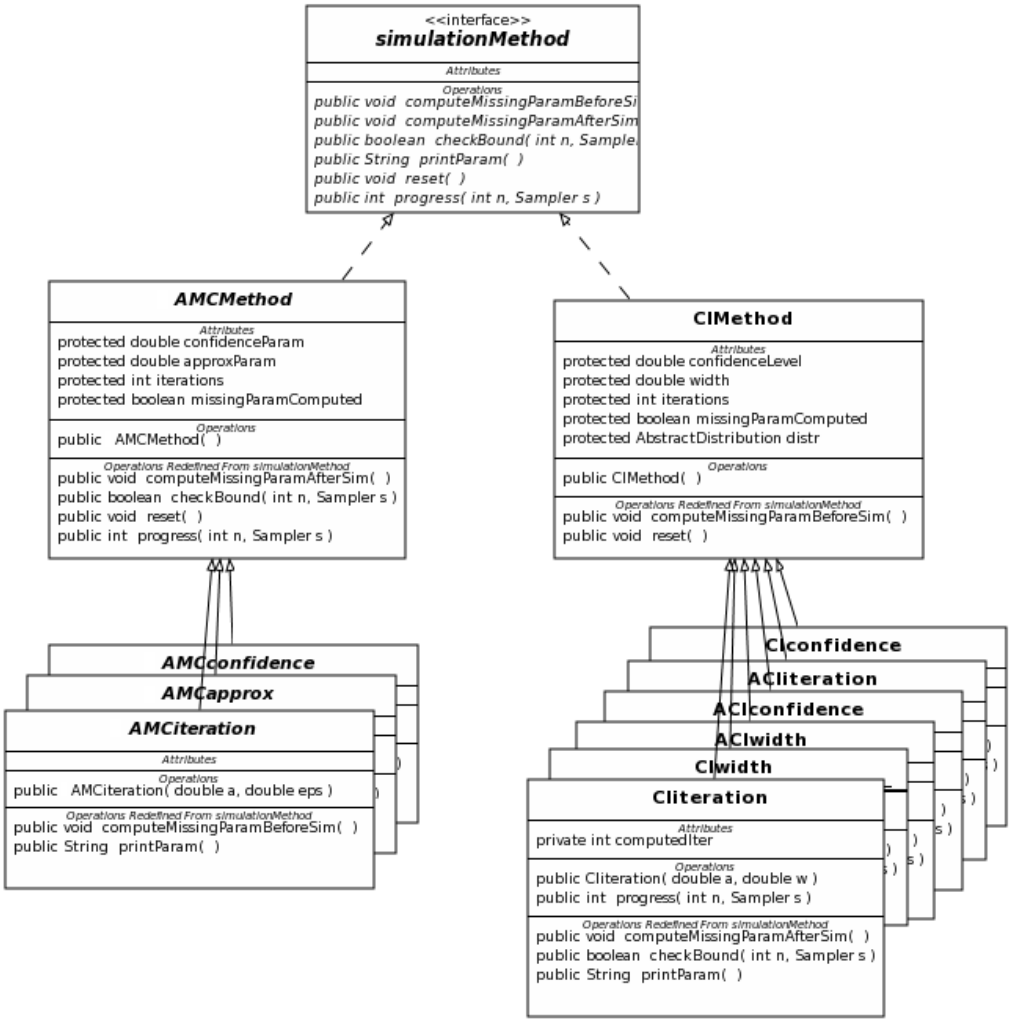


Figure 3.15: UML diagram of the implementation

Note that the missing parameter computation before simulation does nothing in case of CI or ACI method (because they require the simulation to estimate the last parameter), and the missing parameter computation after simulation is also null in case of AMC method (because the last parameter can be computed independently from the simulation). At any time, we can print all the (known) parameters.

3.7 Summary

We introduced in this chapter the CI, ACI and AMC methods, which can compute an estimation of the property's probability. CI and ACI re-estimate the bound at every iteration, *i.e.* at every time they pick a new path, and need the optimal number of iterations. They can not however compute the last parameter of the problem before or during the simulation. AMC can compute the missing parameter before the simulation, and can fix the number of iterations before the simulation. Unfortunately, it relies on a probability upper bound, which can be large, and the size of sample it needs can be much higher than the real required one. This induces that the confidence parameter is greater than the real computed level of confidence.

If AMC is valid whatever the values are, CI and ACI methods assume the variance is not null. So one have to choose whether we are in a case of null variance or not. The decision maker we propose, based on a (arbitrary) simple observation, offers good results for general cases (including the examples we used). If we have a model which is likely to have a property always true or false, *i.e.* which can have a null variance, then it is wiser to specify manually the number of iterations after which we can conclude about the result.

These methods have been implemented in the console and graphic modes of PRISM (to use them, read appendix A).

Chapter 4

Statistical Model Checking: $R_{=?}[\phi]$ problem

In this chapter, we focus on the estimation of the value of a reward, *i.e.* for formula of type $R_{=?}[\phi]$ with a given reward structure $< \rho, \iota >$. Up to our knowledge, the problem of computing reward expectation statistically is not treated in the literature. Our task here will be then to develop new methods adapted to this problem, inspired by the previous methods for probabilities. In particular, we will see that the method for CI and ACI computation (and all their possible derivations) can almost directly be extended to this problem in section 4.1. This is illustrated by experimentations in section 4.3. Unfortunately, AMC method requires some hypotheses which are not holding in this case anymore. We will then propose two different extensions of the AMC method in sections 4.4 and 4.5 with different assumptions, test them in section 4.6 and compare them practically in section 4.7.

4.1 Confidence interval methods

As for 3.1, we first extract some samples ω_i from the Markov chain. Then, instead of representing the validity of a property P over a run ω by $\chi_p(\omega)$, we represent the reward of this path for a given property P and a reward structure Σ by $\chi_{P,\Sigma}(\omega)$ ($\chi_{P.\Sigma} \in \Omega \longmapsto \mathbb{R}^+$). The $\chi^i_{P,\Sigma}$ are still supposed IID and normally distributed, so we still have:

$$Z' = \frac{\overline{Y} - E[\chi_{p,\Sigma}]}{\sqrt{\frac{S^2}{N}}} \rightsquigarrow T_{N-1}.$$

Note however that Chernoff-Hoeffding bound does not hold anymore with non

Bernouilli variables. We have then no guarantee when using AMC method.

Practically, the implementation is the same as for properties probabilities, except that we can no more use the optimization we were using to compute $\overline{Y}$ and S^2 - we have to compute both of them at every iteration.

The other problems are treated in the same way as for CI and ACI with probabilities in 3.6. The formulae (3.10), (3.11), (3.12) and (3.13) remain valid.

4.2 Decisions

In CI and ACI cases where the number of iterations is not known before the simulation, we still have to decide whether we are in $S^2 = 0$ or $Z' \rightsquigarrow T_{N-1}$ case during a simulation. Once again, we propose an arbitrarily solution based on the width ε:

$$\left| v - \frac{nv + Y_{n+1}}{n+1} \right| \leq \varepsilon, \tag{4.1}$$

where v is a non-negative real, n the number of iterations already done and Y_{n+1} the last realisation which can equal v or not. The decision becomes

$$\begin{aligned} \left| \frac{v - Y_{n+1}}{n+1} \right| \leq \varepsilon &\Leftrightarrow 0 \leq \varepsilon \wedge \frac{|v-w|}{n+1} \leq \varepsilon \\ &\Leftrightarrow n \geq \frac{|v-w|}{\varepsilon} - 1. \end{aligned}$$

where w is the value of Y_{n+1}. We can give to $\frac{|v-w|}{\varepsilon} - 1$ an upper bound[1]:

$$max\left\{ \frac{v}{\varepsilon} - 1, \frac{maxReward}{\varepsilon} - 1 \right\} = max\left\{ \frac{\overline{Y}_n}{\varepsilon} - 1, \frac{maxReward}{\varepsilon} - 1 \right\},$$

which equals

$$\frac{maxReward}{\varepsilon} - 1,$$

because $\overline{Y}_n \leq maxReward$, so any $n > \frac{maxReward}{\varepsilon} - 1$ respects the condition above.

[1] Here *maxReward* refers to the maximum value the reward can get in general.

For DTMCs, we could have then[2]

$$n \geq \frac{maxStructureReward \times maxPath}{\varepsilon} - 1.$$

For CTMCs with bounded time properties, the decision could take the form

$$n \geq \frac{maxStructureReward \times maxTime}{\varepsilon} - 1.$$

In PRISM, as for the probabilities, we let the user the possibility of providing manually the number of samples from which we can make a decision.

4.3 Confidence interval experiments

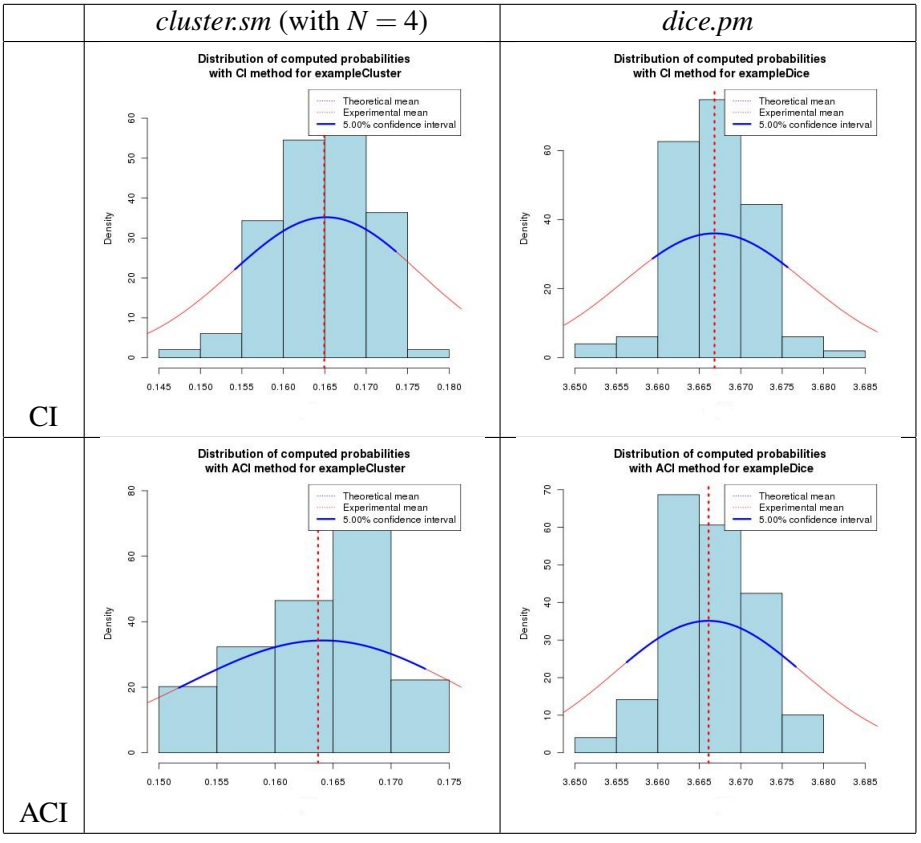


Figure 4.1: Comparisons of distributions for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM.

[2] *maxStructureReward* refers to the maximum value existing in the structure.

We compare *cluster.sm* and *dice.pm* examples in terms of

- distribution of computed estimation of expectation for 100 runs in figures 4.1;
- time and number of samples required in figure 4.3;
- distribution of number of samples required for 100 runs in figure 4.2.

The results for CI and ACI methods are correct, and the bold part of the curve is close to the $2w$-width interval centred in the real expectation $E[\chi_{P,\Sigma}]$.

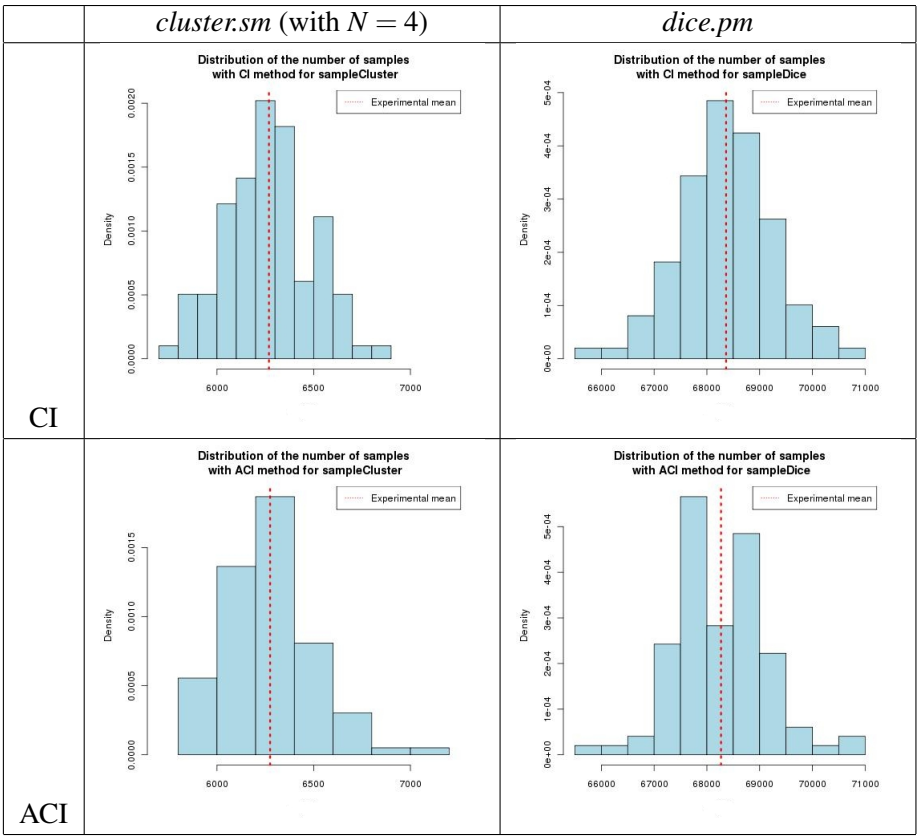


Figure 4.2: Comparisons of distributions of number of samples required for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM.

	cluster.sm (with $N = 4$)	*dice.pm*
CI	6,250 samples / < 1 sec	68,400 samples / 1 sec
ACI	6,250 samples / < 1 sec	68,300 samples / 1 sec
AMC (not valid)	18,445 samples / 1 sec	18,445 samples / 1 sec

Figure 4.3: Comparisons of times for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM.

We observe in figure 4.4 with the dice example that AMC method is no more valid in this context. The bold part of the curve is indeed much greater than the interval centred on the real expectation with width $2w$. This is explained by the fact that the random variables are no more Bernouilli variables, and thus are no more bounded by 1. We will need to modify this method.

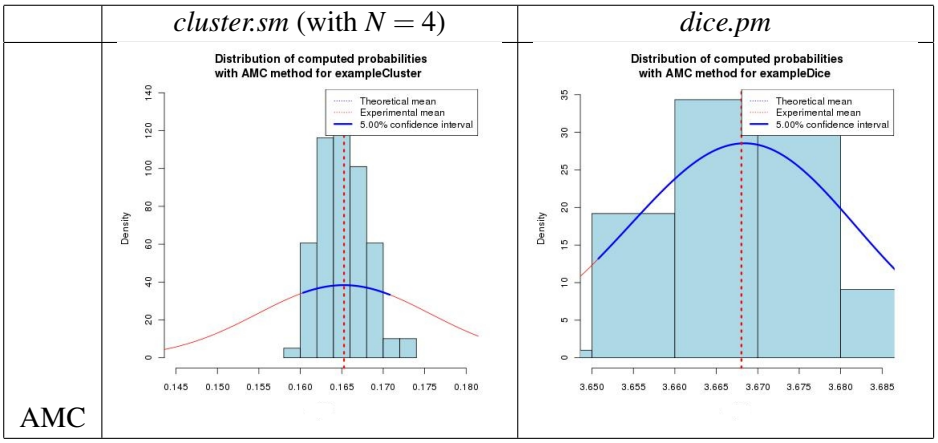


Figure 4.4: Comparisons of distributions for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM with normal AMC method.

4.3.1 Limit test

An interesting limit test for this Monte Carlo method would be to have the $\chi_{P,\Sigma}$ variable taking extreme values. For instance, there would be a path with very low probability in the Markov chain[3] inducing a very high reward whereas the reward would be null everywhere else.

For example, let us consider a modification of the dice Markov chain in figure

[3]From the Monte Carlo point of view, there would be few paths ω_i in Ω such that $\chi_{P,\Sigma}(\omega_i) >> 0$.

4.5, with the reward structure

$$\begin{cases} \rho = 0 \\ \iota(s,d) = 100,000 \text{ if } s = *, d = 2 \\ \iota(s,d) = 0 \text{ otherwise} \end{cases} .$$

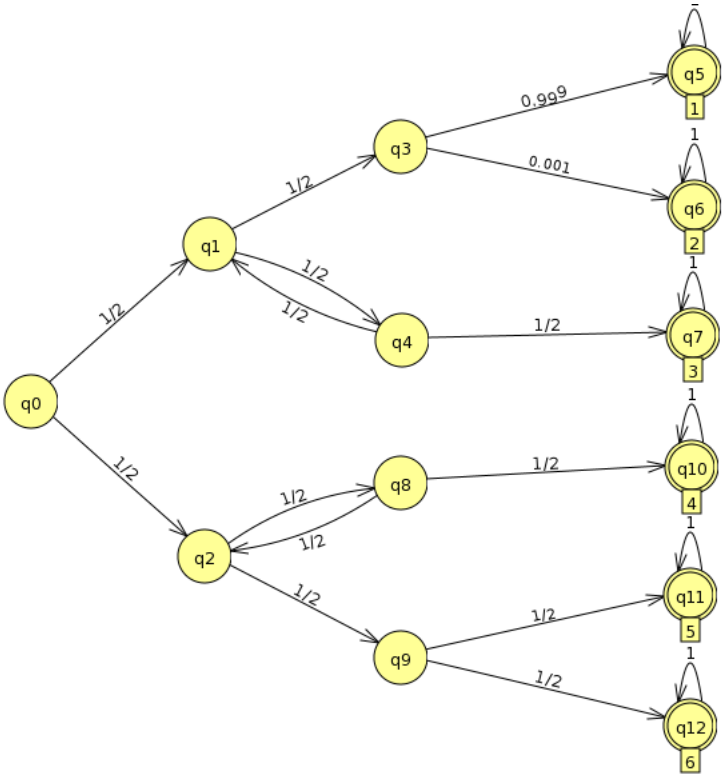


Figure 4.5: Modified Markov chain of the die

We compute the reward for the following property: $R_{=?}[F\ s = 8]$ (numerical result: 33.3).

For $\alpha = 5\%$, width$= 1$, CI method computes the correct CI in $N = 12671165$ samples, in 131 seconds on the test configuration.

AMC methods do not work for this example - for a CI supposed to be with $\varepsilon = 0.01$, we generally have an ε between 1 and 10.

The CI and ACI methods without the number of iterations known before the simulation require however the $S^2 = 0$ against $Z' \rightsquigarrow T_{N-1}$ cost/reward decision - the decision for probabilities used before would consider we are in a $S^2 = 0$ case, given the width and the very low probability[4]. This minimal number of iterations required to conclude is here given manually - $N' = 1000000$ in this example.

4.4 Bounded extension of AMC

A nice property of AMC method is that we can compute the missing parameters without any simulation.

We would like to extend the AMC method to cost/reward system - the AMC required indeed to have Bernouilli variables (i.e. variables with values in $\{0,1\}$), whereas here the random variables are in $\Omega \longmapsto \mathbb{R}^+$. [12] extends the bound to any bounded random variables. So, if we know the maximum reward *maxReward* which can be computed in a model, we have

$$Prob[|A-\gamma| > \varepsilon] < 2e^{-\frac{2N\varepsilon^2}{maxReward^2}}.$$

We derive then

$$\delta = 2e^{-\frac{2N\varepsilon^2}{maxReward^2}}, \tag{4.2}$$

and the other equations similarly to the AMC method in 3.3:

$$N = \frac{maxReward^2}{2\varepsilon^2} \times ln\left(\frac{2}{\delta}\right), \tag{4.3}$$

$$\varepsilon = \sqrt{\frac{maxReward^2}{2N} \times ln\left(\frac{2}{\delta}\right)}. \tag{4.4}$$

Note that we still have a precondition: $\delta = 2e^{-\frac{2N\varepsilon^2}{maxReward^2}} \leq 1$, which leads to

$$N\varepsilon^2 \geq \frac{maxReward \times ln2}{2}. \tag{4.5}$$

[4] Or, from Monte Carlo point of view, the rareness of path offering a non null reward.

4.4.1 What happens if the bound is exceeded?

Suppose that among the realizations Y_i, there is a j such that $Y_j > maxReward$. This means that there is a $realMaxReward > maxReward$. We compute with a level of confidence $\alpha = 2e^{-\frac{2n\varepsilon^2}{maxReward^2}}$, which is then lower than the real level $\alpha_{real} = 2e^{-\frac{2n\varepsilon^2}{realMaxReward^2}}$ we should compute. So the confidence interval we have computed, CI_{AMC}, is smaller than the one we think we have computed, say CI^*. In other words, if the means are centred, we have $CI_{AMC} \supseteq CI^*$, which is not convenient. The (unpredictable) α increases directly with the distance between *maxReward* and *realMaxReward* (if $maxReward < realMaxReward$).

This can be illustrated with figure 4.6. The real reward of *dice.pm* is 3.666. The graphs have been computed for $\varepsilon = 10^{-2}$ and $\delta = 5\%$, the left graph with $maxReward = 1.0$ (< 3.66), the right one with $maxReward = 4.0$ (≥ 3.666). It is clear the confidence interval on the left is bigger than the one expected (3.666 ± 0.01), whereas the right one is smaller.

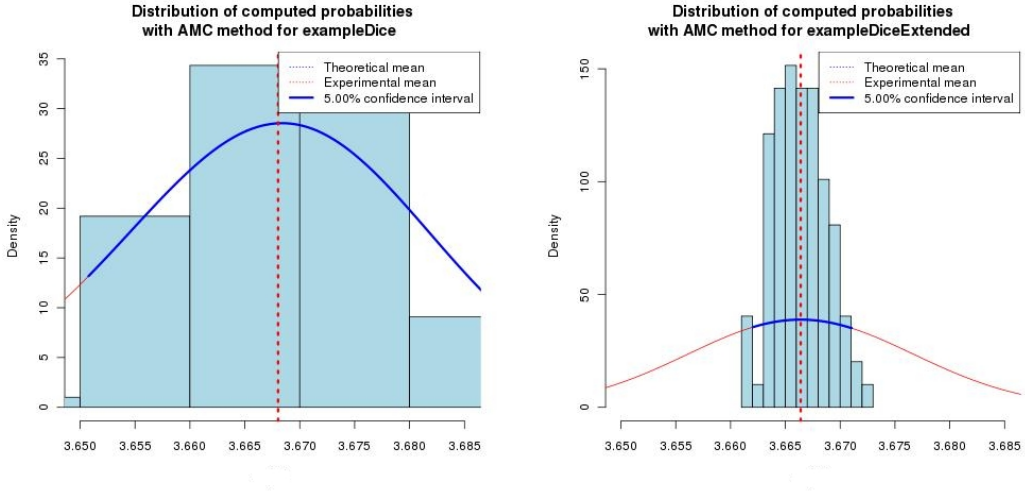


Figure 4.6: Comparison of distributions of *dice.pm* rewards for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM, with extended AMC method. The maximum reward for the left graph is 1.0, the right graph's one is 4.0 (real reward is 3.666).

4.4.2 Implementation

Figure 4.7 gives a decision tree to find out which additional parameter must be given to make a computation.

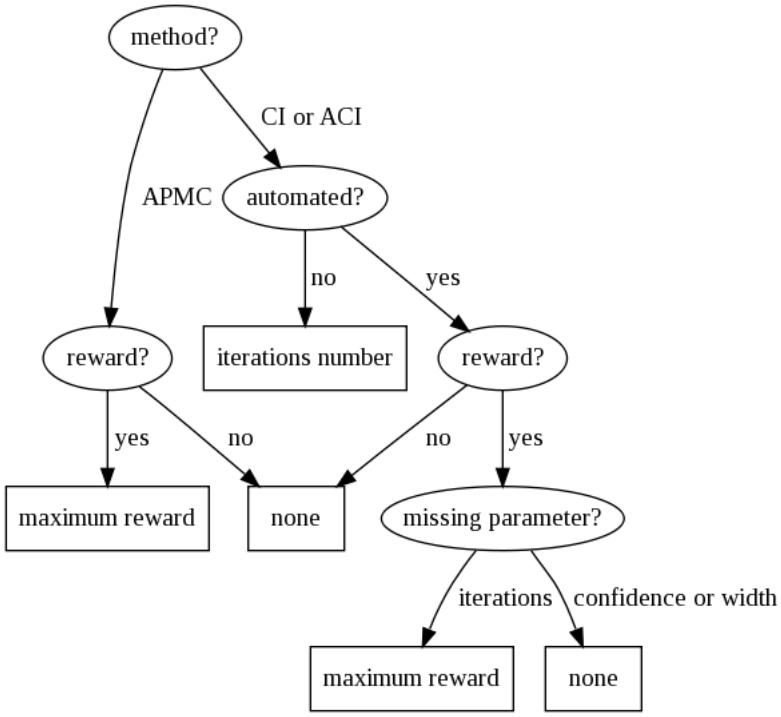


Figure 4.7: Decision tree to find which additional parameter to use.

4.4.3 Limit test

In the previous example, requiring about $N = 12,671,165$ samples with CI and ACI methods, AMC method would require $\frac{100000^2}{2\times 1} \times ln\left(\frac{2}{0.05}\right) \approx 1.84 \times 10^{10}$ iterations, which is out of the range of this implementation[5] - and would certainly take more than 50 hours of computation on the testing configuration. So we consider the same model, but with a more modest reward structure:

$$\begin{cases} \rho = 0 \\ \iota(s,d) = 10,000 \text{ if } s = *, d = 2 \\ \rho(s,d) = 0 \text{ otherwise.} \end{cases}$$

We get then the correct results after the numbers of iterations presented in figure 4.8.

[5] We encoded iterations with Java integer, which cannot be greater than $2^{31} - 1$.

AMC	184,443,973 / 1876 sec
CI	114,268 / 2 sec
ACI	139,944 / 2 sec

Figure 4.8: Number of iterations (and time on the testing configuration) to approximate $R_{=?}[F\ s=8]$

4.5 Unbounded extension of AMC

4.5.1 Unbounded problem

Now, it is not always easy to know the maximum reward. It would be better to extend these formulae to unbounded variables. The formulae developed in [12] are unfortunately limited to bounded variables. In its introduction, [34] suggests that the *truncation method* is a possible method to solve the unbounded problem[6]. We can find this method, combined with Markov's inequality, in [30] for example. We adapt this method to our context.
Let us consider the random variable

$$V := \frac{1}{n}\sum_{i=1}^{n} X_i - E[\chi_{P,\Sigma}].$$

We split it into two random variables according to a value L:

$$V =: V_{\leq L} + V_{>L}.$$

For the bounded variable, we can apply the general Hoeffding's inequality:

$$Prob\left(|V_{\leq L}|\right) = Prob\left(\left|\left(\frac{1}{n}\sum_{i=1}^{n} X_i - E[\chi_{P,\Sigma}]\right)_{\leq L}\right| \geq \varepsilon\right) \leq 2e^{-\frac{2n\varepsilon^2}{(L+E[\chi_{P,\Sigma}])^2}},$$

because $X_i - E[\chi_{P,\Sigma}] \leq L$.
For the unbounded part, we notice that, if $L \geq \varepsilon$,

$$Prob\left(|V_{>L}| \geq \varepsilon\right) = Prob\left(V > L \cap V_{>L} \geq \varepsilon\right) + \underbrace{Prob\left(V \leq L \cap 0 \geq \varepsilon\right)}_{=0}$$

$$\leq Prob\left(V > L \cap \underbrace{V_{>L} + V_{\leq L}}_{=V} \geq \varepsilon\right) = Prob(V > L) \leq Prob(V \geq L).$$

[6] Even though it does not study it.

We apply Markov's inequality and get

$$Prob\left(|V_{>L}| \geq \varepsilon\right) \leq Prob(V \geq L) = Prob(\frac{1}{n}\sum_{i=1}^{n} X_i \geq L + E[\chi_{p,\Sigma}]) \leq \frac{E[\chi_{P,\Sigma}]}{L + E[\chi_{P,\Sigma}]}.$$

By summing, we get finally

$$Prob(\left|\overline{X} - E[\chi_{P,\Sigma}]\right| \geq \varepsilon) \leq 2e^{-\frac{2n\varepsilon^2}{(L+E[\chi_{P,\Sigma}])^2}} + \frac{E[\chi_{P,\Sigma}]}{L + E[\chi_{P,\Sigma}]}. \tag{4.6}$$

If $L \to +\infty$, we recognize the bound from 4.4, i.e.

$$2e^{-\frac{2n\varepsilon^2}{(L+E[\chi_{P,\Sigma}])^2}} + \frac{E[\chi_{P,\Sigma}]}{L + E[\chi_{P,\Sigma}]} \sim 2e^{-\frac{2n\varepsilon^2}{L^2}}.$$

But we notice that a moment of the random variables appears in the bound. This can be estimated with the realizations of the random variables, but only after the simulation - we lose the property of computing missing parameters without simulation.

Note that we are also adding a new parameter, L, which can be considered as the supposed maximum reward - but without requiring to be the real one. As suggested in [30], we can also choose an arbitrary[7] $\gamma > 0$, and take L such that

$$E\left[\chi_{P,\Sigma}\right] \leq \gamma.$$

[34] proposes a more theoretical method to deal with this unbounded problem. Unfortunately, this requires the knowledge of another random variable U which dominates X, i.e. $\forall v, Prob(X < v) \leq Prob(U < v)$. So we still have a kind of arbitrary bound.

We can find in [22] the Bernstein's inequality presented as a generalization of Hoeffding's bound.

$$\begin{cases} P\left[n(\overline{X} - E[\chi_{P,\Sigma}]) \geq \sqrt{2v}x + cx\right] \leq e^{-x} \\ \text{with } nE[\chi^2_{P,\Sigma}] \leq v \\ \text{and } \forall k \geq 3, nE[(\chi^k_{P,\Sigma})_+] \leq \frac{k!}{2} v c^{k-2}. \end{cases}$$

But this inequality, which does not require any bound on random variables, has some moments of the random variables in the bound. We can estimate them, but

[7] It is represented by δ in the [30].

one more time, it also means we can compute the missing parameters only at the end of the simulation.

In terms of interpretation, we were looking here for a bound which, for an α and a ε given, was the same whether the maximum reward was 1 or 10^{12}. This objective is maybe a bit optimistic in this general case.

4.5.2 Unbounded problem with bounded expectation

We assumed from the beginning the expectation was finite, i.e. $E[\chi_{P,\Sigma}] \leq M$, for some $M > 0$. If we know a γ such that $\gamma \geq M$, and because we also know that $E[\chi_{P,\Sigma}] > 0$, the bound of equation (4.6) has for upper bound

$$\delta = 2e^{-\frac{2n\varepsilon^2}{(L+\gamma)^2}} + \frac{\gamma}{L}, \tag{4.7}$$

with $L \geq \varepsilon$ a real number. Because of the exponential and the fact that δ and $\frac{\gamma}{L}$ are > 0, we get the condition

$$L > \frac{\gamma}{\delta}.$$

So, given γ, ε and n, we can compute δ (for some L). Conversely, if we are given γ, ε and δ, we can compute n with

$$n = \left\lceil \frac{(L+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2L}{\delta L - \gamma}\right) \right\rceil, \tag{4.8}$$

and ε with

$$\varepsilon = \sqrt{\frac{(L+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2L}{\delta L - \gamma}\right)}, \tag{4.9}$$

for some L. Now our objective is to get the minimum n, so we have to choose the $L > \frac{\gamma}{\delta}$ which minimizes

$$f(x) = \frac{(x+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2x}{\delta x - \gamma}\right).$$

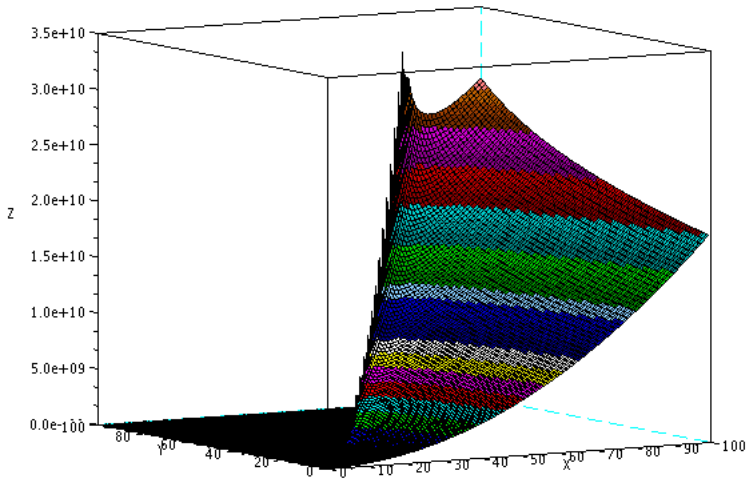


Figure 4.9: $n = \frac{(L+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2L}{\delta L-\gamma}\right)$, with $\delta = 5\%$, $\varepsilon = 10^{-2}$, γ (Y) from 0.5 to 40, L (X) from 10 to 1000, n (Z). The function is not defined on the left part, where $L \leq \frac{\gamma}{\delta}$. The peaks in the middle represent actually an asymptote for $L \to \frac{\gamma}{\delta}$. We can see on the admissible part a local minimum.

As we can see in figure 4.9, this is a hyperbolic function. $\lim_{x\to\frac{\gamma}{\delta}} \frac{2x}{\delta x-\gamma} = +\infty$, so $f(x) \to +\infty$ when $x \to \frac{\gamma}{\delta}$. On the other side, we have[8]

$$\frac{(x+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2x}{\delta x-\gamma}\right) \sim_{+\infty} \frac{x^2}{2\varepsilon^2} ln\left(\frac{2}{\delta}\right),$$

so $f(x) \to +\infty$ when $x \to +\infty$ as well. All the functions composing f are continue and non-null on $]\frac{\gamma}{\delta}, +\infty[$, so there is at least a minimum on this interval. It does not seem easy to find an expression minimizing f, so we compute numerically a L which is very close to L_{min}.

[8] $\frac{2x}{\delta x-\gamma} \sim_{x\to+\infty} \frac{2}{\delta}$, $\frac{2}{\delta}$ is not null and $lim_{x\to+\infty} \frac{2x}{\delta x-\gamma} \neq 1$, so we have

$$ln\left(\frac{2x}{\delta x-\gamma}\right) \sim_{x\to+\infty} ln\left(\frac{2}{\delta}\right).$$

Note that we still have the condition $\delta = 2e^{-\frac{2n\varepsilon^2}{(L+\gamma)^2}} + \frac{\gamma}{L} \leq 1$, leading to

$$n\varepsilon^2 \geq \frac{(L_{min}+\gamma)^2 ln\left(\frac{2L}{L_{min}-\gamma}\right)}{2}. \tag{4.10}$$

4.6 AMCs experiments

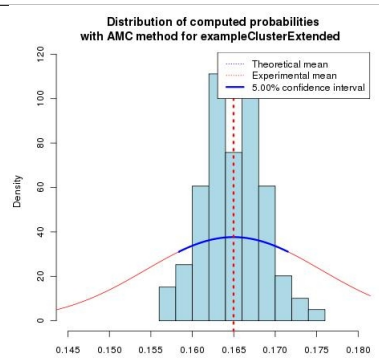
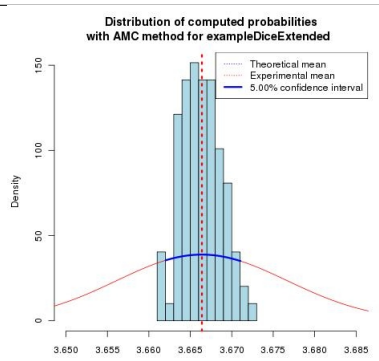
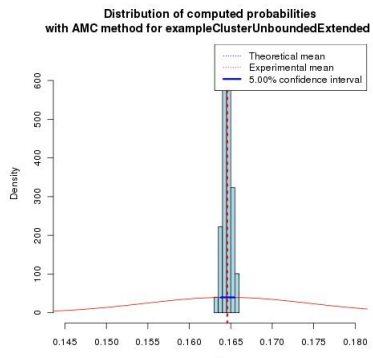
	cluster.sm (with $N = 4$)	*dice.pm*
extended bounded AMC (with $max_i\{X_i\}$=1.0 for *cluster.sm*, 4.0 for *dice.pm*)		
extended unbounded AMC (with $E[X] \leq$0.2 for *cluster.sm*, 3.7 for *dice.pm*)		Not computed (would require around 2.5 days for 100 runs)

Figure 4.10: Comparisons of distributions for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM (for the extended AMC, we took 1.0 for *cluster.sm* and 4.0 for *dice.pm* as maximum reward).

We compare *cluster.sm* and *dice.pm* examples in terms of

- distribution of computed estimation of expectation for 100 runs in figures 4.10;
- time and number of samples required in figure 4.11.

extended bounded AMC	18,445 samples / 1 sec	295,100 samples / 3 sec
extended unbounded AMC	657,109 samples / 17 sec	225,717,722 samples / 35 min

Figure 4.11: Comparisons of times for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, 100 runs of PRISM (for the extended AMC, we took 1.0 for *cluster.sm* and 4.0 for *dice.pm* as maximum reward).

The results are correct, but the level of confidence actually computed is much greater than the wanted one, especially for the unbounded case.

4.7 Bounded vs. unbounded AMC extensions

The unbounded AMC extension is likely to treat correctly more problems than the bounded extension can, because the constraint is lighter: $E[\chi_{P,\Sigma}] \leq \gamma$ instead of $\forall i, X_i \leq maxReward$. A path with an abnormally high reward will not affect the global result of unbounded extension - in this sense, it is more reliable. The constraint of having an upper bound of the expectation seems also natural, because we assume from the beginning that it is finite.

If we consider the limit test from section 4.3.1, almost all the samples produce a null reward, but there is one path, with a very low probability, which produces 100000. The expectation $E[\chi_{P,\Sigma}]$ approximatively equals 3.33, so we can fix γ to 3.5, for instance, and get the result after $201,972,770$ samples (35min with the test configuration). With the bounded method, we need to fix $maxReward = 100000$ to get a correct result, and, as presented in subsection 4.3.1, this would require around 1.84×10^{10} iterations, which is out of the range of our implementation.

On the other hand, the probability bound of the unbounded method is growing clearly faster than the bounded one. If we consider the *cluster.sm* example (table 4.3), for $maxReward = 1.0$, bounded method requires $18,445$ samples whereas for $\gamma = 0.2$, unbounded method needs $657,109$ samples. The gap is really getting bigger with *dice.pm*, where the bounded method uses $295,100$ samples for $maxReward = 4.0$ and the unbounded one produces $225,717,722$ samples for only

$\gamma = 3.7$. The unbounded method appears to be less efficient.

We can ask when it is better to use one or the other method. In this perspective, we compare the respective numbers of iterations:

the unbounded method requires more iterations iff

$$\frac{(L_{min}+E[\chi_{P,\Sigma}])^2}{2\varepsilon^2} ln\left(\frac{2L_{min}}{\delta L_{min}-E[\chi_{P,\Sigma}]}\right) \geq \frac{max_i^2\{X_i\}}{2\varepsilon^2} ln(\frac{2}{\delta}),$$

or

$$(E[\chi_{P,\Sigma}]+L_{min})\sqrt{\left|\frac{ln\left(\delta-\frac{E[\chi_{P,\Sigma}]}{L_{min}}\right)}{ln(\frac{2}{\delta})}\right|} \geq max_i\{X_i\},$$

with L_{min} the value minimizing $f(x) = \frac{(x+E[\chi_{P,\Sigma}])^2}{2\varepsilon^2} ln\left(\frac{2x}{\delta x - E[\chi_{P,\Sigma}]}\right)$.

For $\delta = 5\%$ and $\varepsilon = 10^{-2}$, we compute the relation between $E[\chi_{P,\Sigma}]$ and $max_i\{X_i\}$ (figure 4.12). If the pair $(E[\chi_{P,\Sigma}], max_i\{X_i\})$ of the problem is above the line, it is better to take the unbounded method - if not, the bounded method will be more convenient.

In other words, if the variance of the data is very high, unbounded method should be used; otherwise, bounded method with a given $max_i\{X_i\}$ should be preferred.

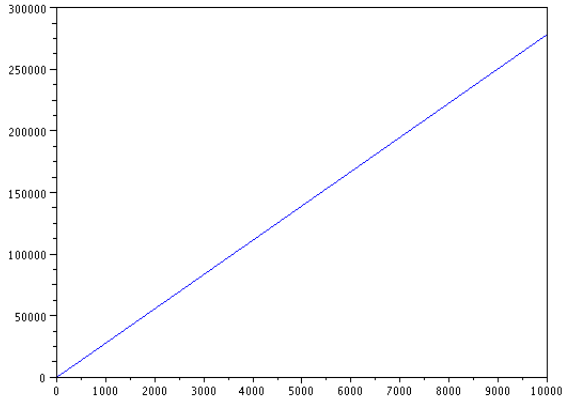


Figure 4.12: $(E[\chi_{P,\Sigma}]+L_{min})\sqrt{\left|\frac{ln\left(\delta-\frac{E[\chi_{P,\Sigma}]}{L_{min}}\right)}{ln(\frac{2}{\delta})}\right|} \geq max_i\{X_i\}$, with $E[\chi_{P,\Sigma}]$ as abscissa and $max_i\{X_i\}$ as ordinate.

4.8 Summary

The objectives of the chapter was to extend the methods of the chapter 3 to rewards computation. If mathematically the CI and ACI methods are almost identical to previously, we showed theoretically and experimentally that AMC method can not be applied anymore. On the basis of the same Hoeffding's bound, a first solution was proposed, but it assumes that we have the knowledge of the maximum reward a path picked by lot can get, which is rarely obvious - and a too small bound induces an uncontrolled unreliability in the parameters. Using truncation method, Markov's and Hoeffding's inequalities, we ended up with a second solution, which assumes we know a bound of the expected value we want to compute. This is consistent and more natural, because we assume from the beginning the expectation of the variable was finite.

Experimentally, even through they can compute estimations that the classic numerical model checker could not, these two methods really suffer from their distance from the optimal number of iterations (given by the CI method, for instance), and the effect is strongly amplified by the high expected values of computed rewards, especially for the second one. We compared the required numbers of iterations, and conclude that the second method has to be used in case of high variance in the validity evaluation of the paths. This is confirmed experimentally by our set of tests and a limit test we designed.

These methods have been implemented in the console and graphic modes of PRISM (to use them, read appendix A).

Chapter 5

Statistical Model Checking: $P_{\bowtie\theta}[\phi]$ problem

In this chapter, we turn to the problem of checking property with bounded probabilities: formula of type $P_{\bowtie\theta}[\phi]$. There is two ways of testing this. The first one consists in computing an estimation of the probability with one of the methods described in the previous chapters, and making a decision in terms of this result and the given parameters (section 5.1). Experiments illustrate this in section 5.2. The second one, proposed in [33], relies on the power of the statistical tests - and in particular with the Wald's *sequential probability ratio test* (section 5.3, followed by experimentations in section 5.4). Even though the two approaches require different parameters and seem to work differently, we can sketch some relations between them in section 5.5.

5.1 Using confidence interval methods

In this problem, as the CI computation problem, we are given two parameters among the approximation ε, the confidence level δ and the number of samples N. We aim at checking whether the property is true or not, with a certain level of confidence and an approximation. [11] proposes[1] for the AMC method:

$$P_{\geq\theta}[\phi] \Leftrightarrow \overline{Y}_n \geq \theta - \varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil,$$

[1] [11] proposes actually only the right to left implication, because they work on an unbounded logic (LTL), and they have to bound arbitrarily their model checking. We are working here only on bounded formulae, so this is an equivalence.

so we just have to test the right hand formula[2]. On figure 5.1, this means we accept the null hypothesis (*i.e.* the property is true) if $\overline{Y}_n$ is in the right area or in the grey region.

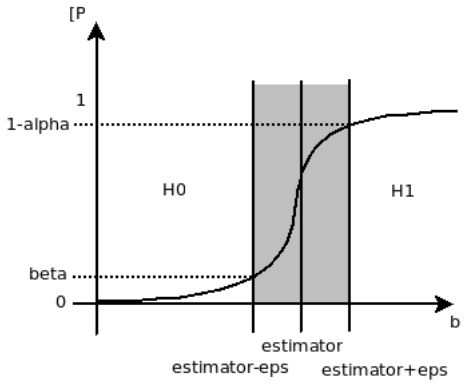


Figure 5.1: Accepting regions for AMC test

We have similarly

$$P_{\leq\theta}[\phi] \Leftrightarrow \overline{Y}_n \leq \theta + \varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil.$$

However, this grey region would clearly define the indifference region: we know that the real probability of the formula, say p^*, is in this area (with a probability δ), but it is possible that this p^* is close to $\theta - \varepsilon$, and if we take a θ close to $\theta + \varepsilon$ would wrongly accept it (type I error).

This is designed to return an answer after every simulation, and to avoid a "unknown" response[3]. Nevertheless, this gives advantage to the null hypothesis, and testing $P_{\leq\theta}[\phi]$ is no more equivalent to $\neg P_{>\theta}[\phi]$. That is why we will prefer an indifference region in our implementation, to remain consistent with the SPRT approach we will present in the section 5.3 - a comparison between the two methods will be proposed in section 5.5.

So if we are in the H_0 accepting region, the formula is said true, if we are in the H_1 accepting region, the formula will be considered false, and if we are in indifference region, the user will be advised to either increase or decrease its θ, or

[2]Here, we assume that $>\approx\geq$ and $<\approx\leq$, which is an acceptable approximation given the numerical truncations in computation and the ε in the formula.

[3]APMC, which implements this choice, has indeed been thought to always provide an answer - and preferably a quantitative one, through estimations.

to decrease the ε, which clearly reduces the size of this region (but also increases the samples size).

In other words,

$$P_{\geq p}[\phi] \Leftarrow \overline{Y}_n \geq p+\varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil, \tag{5.1}$$

$$P_{\leq p}[\phi] \Leftarrow \overline{Y}_n \leq p-\varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil, \tag{5.2}$$

and $p-\varepsilon \leq \overline{Y}_n \leq p+\varepsilon$ means that we have to refine the parameters.

5.2 Confidence interval experiments

The figure 5.2 presents the experimentation of these techniques with the three biological examples. The number of iterations and times are the same as in table 3.11, and do not depend on the value of θ[4].

Every figure is a zoom on the values θ in abscissa which are close to the real probability. Each dot represents on ordinates the percentage of *true* answers by the model checker for $\theta = x$ in abscissa, with properties of form $P_{\leq\theta}[\phi]$. To be clearer, we choose for this experimentation not to consider the indifference region and to advantage the H_0 hypothesis, as APMC does. We observe, as expected, a transition between $false$ and *true* around the real probability. With our implementation, the transition could be not as visible as it is here - the transition mostly lies in the indifference region. The figure 5.3 presents the kind of graph we obtain with our normal implementation.

[4]Because $\overline{Y}_n$ is computed first, and then we only compare the result to θ

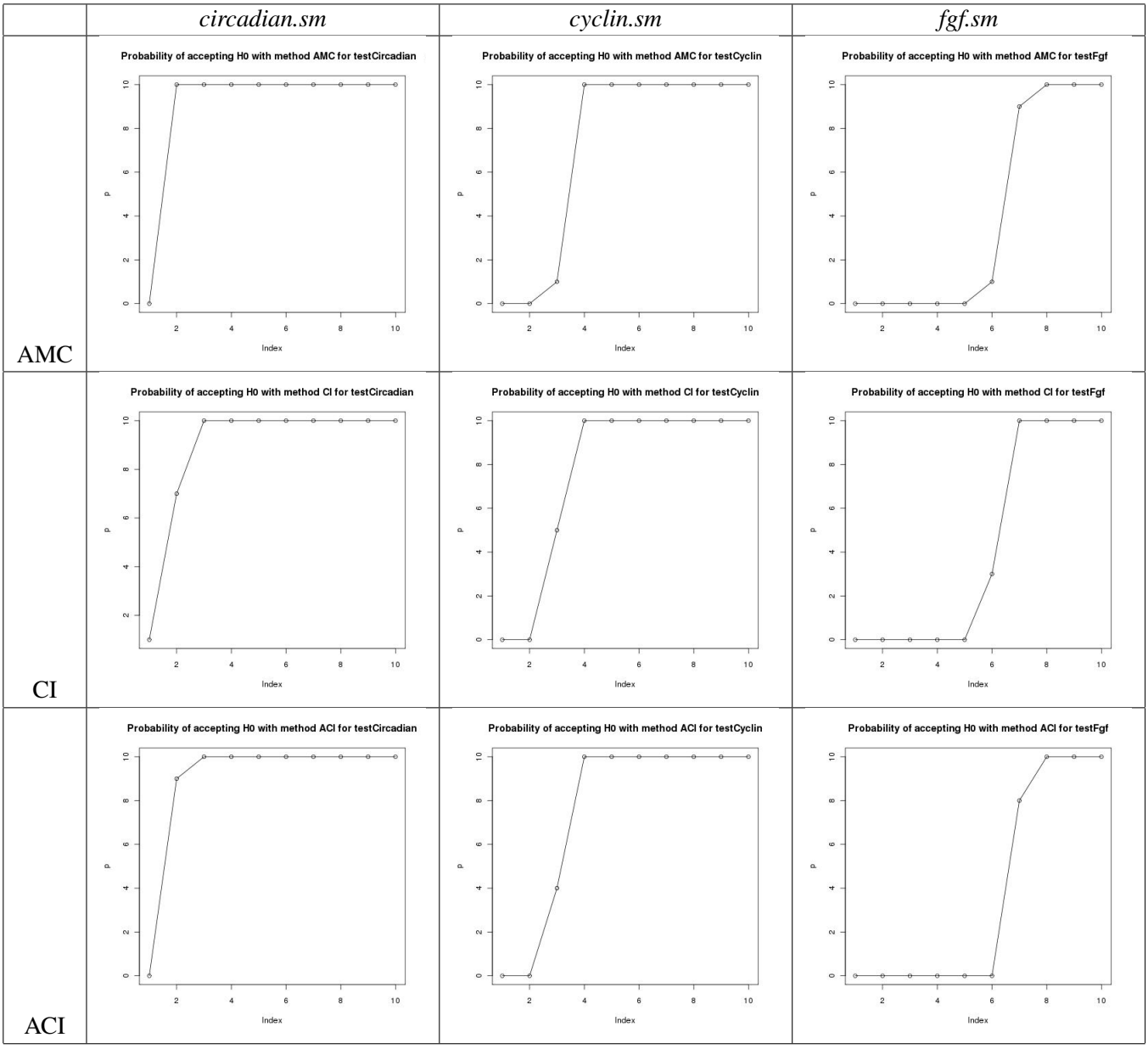


Figure 5.2: Comparisons of the probabilities of accepting H_0 hypothesis (ordinates), for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa), without indifference region.

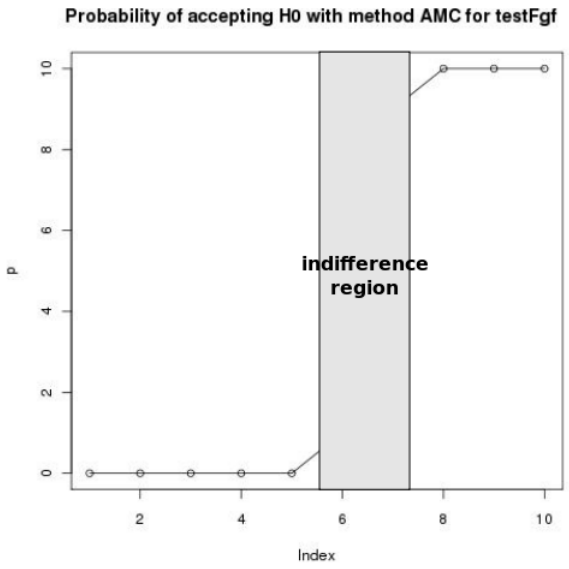


Figure 5.3: Example of probabilities of accepting H_0 hypothesis (ordinates), for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa), with indifference region.

5.3 Using statistical test

If we want to check $P_{\geq\theta}[\phi]$, [33] proposes to use the statistical test

$$\begin{cases} H_0 : p \geq \theta + \delta' =: p_0 \\ \text{against} \\ H_1 : p \leq \theta - \delta' =: p_1 \end{cases}, \tag{5.3}$$

for some $\delta' > 0$, with α_e the type I error and β_e the type II error. We can use the Wald's *sequential probability ratio test*, which compares the likelihoods ratio of the binomial distribution under H_0 and H_1 hypotheses

$$\frac{L(x_1,\ldots,x_m,p_1)}{L(x_1,\ldots,x_m,p_0)} = \frac{p_1^{d_m}(1-p_1)^{m-d_m}}{p_0^{d_m}(1-p_0)^{m-d_m}}, \tag{5.4}$$

where $d_m = \sum_{i=1}^{m} x_i$, to two values A and B:

- if the ratio is lower than B, H_0 is accepted;
- if the ratio is greater than A, H_1 is accepted;
- otherwise, we get a new realization x_{m+1} and iterate.

Practically, we choose $A = \frac{1-\beta_e}{\alpha_e}$ and $B = \frac{\beta_e}{1-\alpha_e}$ [33].

5.4 SPRT experiments

The figure 5.4 presents the experimentation of these techniques with the three biological examples. The number of iterations required by SPRT depends on θ, so these numbers are displayed in figure 5.5 in terms of θ.

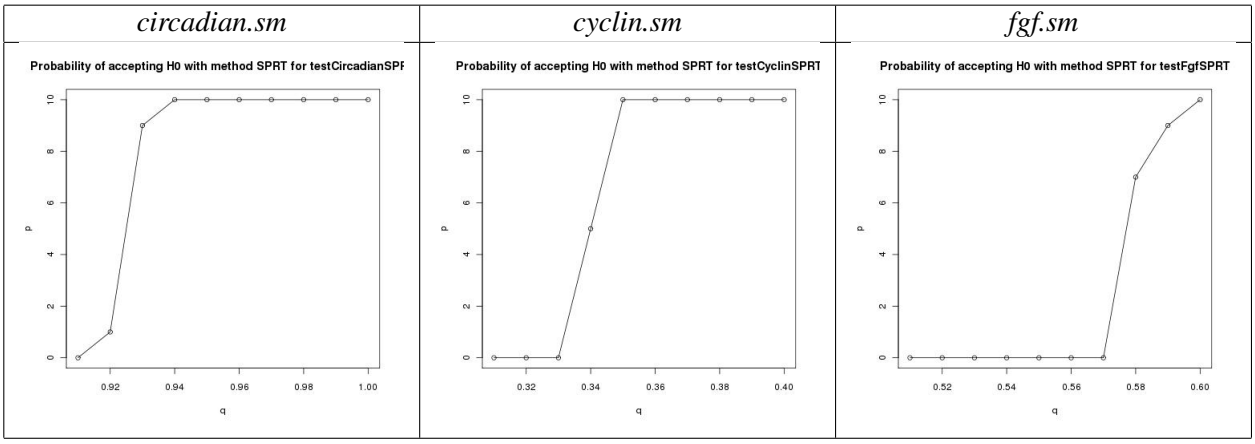


Figure 5.4: Comparisons of the probabilities of accepting H_0 hypothesis (ordinates) with SPRT, for $\alpha_e = 0.1$, $\beta_e = 0.1$, $\delta' = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa).

The closer we are to the real probability, the harder it is for the algorithm to "make the decision", that is to say to have the likelihood growing and accepting H_0 or diving and accepting H_1. This explains the peaks of samples in the neighbourhood of the real probability.

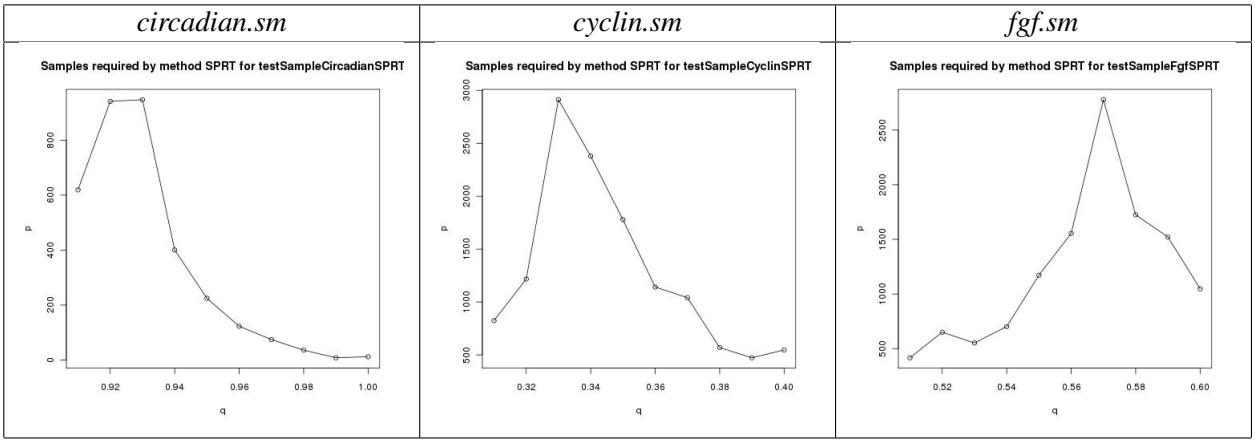


Figure 5.5: Average number of iterations required by SPRT in terms of θ. SPRT is run with $\alpha_e = 0.1$, $\beta_e = 0.1$, $\delta' = 10^{-2}$, and 10 runs are used to compute each point of the graph.

5.5 Relations between CI methods and SPRT

These two methods presented previously have different parameters and at first sight seem to work differently.

AMC method[5] requires two parameters among

- δ, the confidence parameter, which represents the reliability of the computations;
- ε, the approximation parameter, which will define the size of the indifference region;
- N, the number of iterations;
- θ, the threshold probability.

The property is true if we are in H_0, false if we are in H_1, and *unknown* if we are in the indifference region.

SPRT method requires:

- α_e, the type I error, *i.e.* accepting H_0 while H_1;
- β_e, the type II error, *i.e.* accepting H_1 while H_0;

[5] And also CI and ACI methods, with the correspondences of parameters from table 3.2.

- δ', a parameter to make H_0 and H_1 independent, by creating an indifference region;
- N, the number of iterations;
- θ, the threshold probability.

If the likelihood of the distribution is greater than a value $A(\alpha_e, \beta_e)$, H_0 is accepted, if it is lower than $B(\alpha_e, \beta_e)$, H_1 is accepted. Otherwise, we iterate one more time to make the $\overline{Y}_N$ evolves, and so on until we converge.

An iteration of SPRT can be seen as applying AMC method: if we are in the indifference region at iteration m, we iterate, which makes the ε decrease, and thus the indifference region reduces.

So these two methods work globally in the same way, but provide different controls:

- SPRT controls α_e and β_e, and this define $A(\alpha_e, \beta_e)$ and $B(\alpha_e, \beta_e)$ and thus the width of the indifference region at every iteration;
- CI methods control directly the width of the indifference region, through ε, which defines implicitly the α_e and β_e of the test.

5.6 Summary

Tests of properties with threshold probabilities have been investigated in this chapter. We proposed two ways of dealing with this problem. The first one relies on the previous methods from chapter 3: we compute an estimation, and we compare it with the threshold probability, given the parameters. It provides a reliable response, but can answer it cannot conclude, because the estimation is in an indifference region. The user has then to repeat the simulation with refined parameters. The number of iterations for the estimation is optimal for the given parameters with CI method, and it is independent from the threshold probability, but we cannot be sure the test will provide an answer after the simulation.

The second solution is the Wald's sequential probability ratio test, which relies on different parameters. It always converges and provides a reliable answer, and is optimal in terms of iterations for the given parameters. The number of iterations depends however on the threshold probability, and the closer the threshold and real probabilities, the more expensive this algorithm is likely to be.

We finally related the meanings and the operations of the two algorithms and their respective parameters.

These methods have been implemented in the console mode of PRISM (to use them, read appendix A).

Chapter 6

Statistical Model Checking: $R_{\bowtie\theta}[\phi]$ problem

In this chapter, we focus on the statistical model checking of properties with bound rewards, *i.e.* properties of form $R_{\bowtie\theta}[\phi]$. The first possibility is almost identical to the first method described in the previous chapter: we estimate the expectation with a corresponding method from chapter 4 and make a decision, given the parameters (section 6.1, experimentation in section 6.2). However, the second one is no more valid in this context: the distribution the random variables follow is a normal law, with several parameters, and Wald's test does not deal with this. We propose to use the Bartlett's test instead in section 6.3, and experiments confirm this method in section 6.4.

6.1 Using confidence interval methods

We use the same technique as in 5.1 (this time with $\theta \in \mathbb{R}_+^*$), with CI and ACI, or the extensions of AMC:

$$R_{\geq\theta}[\phi] \Leftarrow \overline{Y}_n \geq \theta + \varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil, \tag{6.1}$$

$$R_{\leq\theta}[\phi] \Leftarrow \overline{Y}_n \leq \theta - \varepsilon, \text{ with } n = \left\lceil \frac{ln\left(\frac{2}{\delta}\right)}{2\varepsilon^2} \right\rceil, \tag{6.2}$$

and $\theta - \varepsilon \leq \overline{Y}_n \leq \theta + \varepsilon$ means that we have to refine the parameters.

6.2 Confidence interval experiments

The figures 6.1 and 6.2 present the experimentation of these techniques with the *cluster.sm* and *dice.pm* examples. The number of iterations and times are the same as in table 4.3, and do not depend on the value of θ[1].

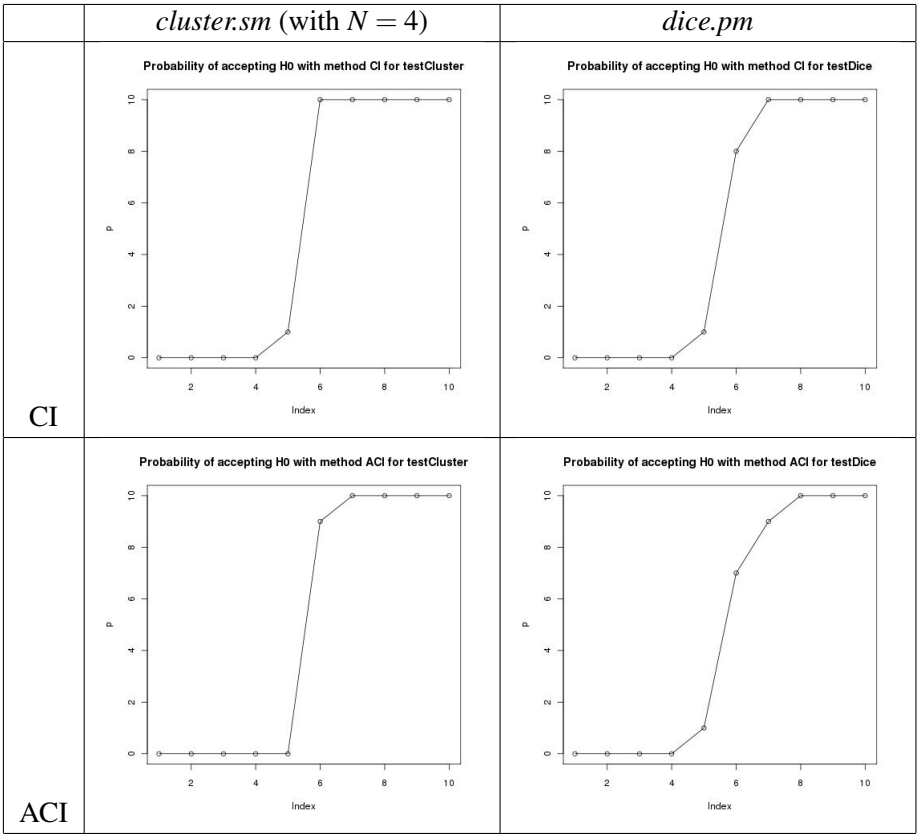


Figure 6.1: Comparisons of the probabilities of accepting H_0 hypothesis (ordinates), for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa).

[1] Because $\overline{Y}_n$ is computed first, and then we only compare the result to θ

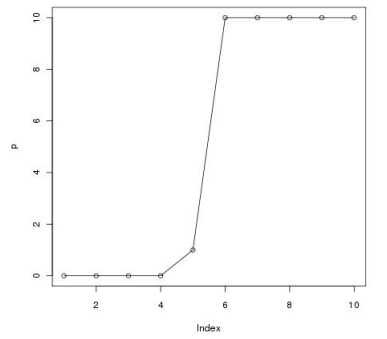
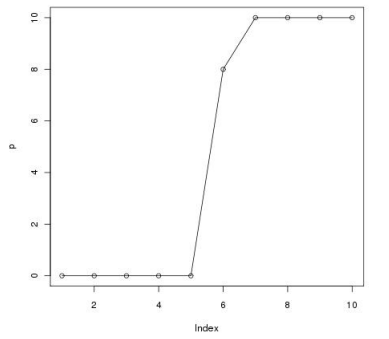
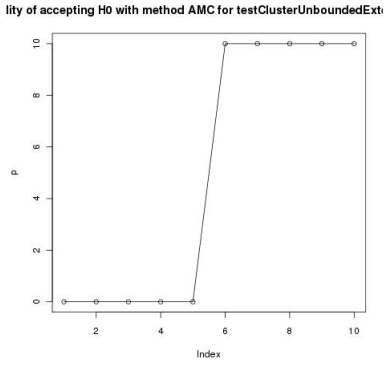
	cluster.sm (with $N = 4$)	*dice.pm*
extended bounded AMC (with $max_i\{X_i\}$=1.0 for *cluster.sm*, 4.0 for *dice.pm*)		
extended unbounded AMC (with $E[X] \leq$0.2 for *cluster.sm*, 3.7 for *dice.pm*)		Not computed (would require around 2.5 days for 100 runs)

Figure 6.2: Comparisons of the probabilities of accepting H_0 hypothesis (ordinates), for $\alpha = 5\%$, $\varepsilon = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa).

6.3 Using statistical test

We can extend the method of section 5.3 to rewards - but this time, we consider normal distributions $\mathcal{N}(\mu_i, \sigma)$, and no more binomial ones. The test would be

$$\begin{cases} H_0 : r \geq \theta + \delta' =: \mu_0 \\ \text{against} \\ H_1 : r \leq \theta - \delta' =: \mu_1 \end{cases}, \tag{6.3}$$

where $\delta' > 0$. Note that the normal distribution has a second parameter, σ^2. We are

not interested in this parameter, but it appears in the likelihood ratio,

$$\frac{L(x_1,\ldots,x_m,\mu_1,\sigma)}{L(x_1,\ldots,x_m,\mu_0,\sigma)} = e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{m}(x_i-\mu_1)^2-(x_i-\mu_0)^2}.$$

and has thus an incidence in the test. This is a *nuisance parameter*, and unfortunately, Wald's SPRT does not support this kind of problems [32].

[29] faced a similar problem in a biostatistics context, and proposed to use the *Bartlett's SPRT* [21], which is really close to the Wald's one, except that it supports distributions with more than one parameter. The strategy consists in replacing the nuisance parameter in both of the conditional likelihoods by its conditional maximum likelihood estimator.

In the specific case of normal distribution, the maximum likelihood estimator of the means is

$$\hat{\mu}_{MLE} = \frac{1}{N}\sum_{i=1}^{N} X_i = \overline{X},$$

and the MLE of variance is

$$\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{i=1}^{N}(X_i-\hat{\mu}_{MLE})^2 = \frac{1}{N}\sum_{i=1}^{N}(X_i-\overline{X})^2,$$

which actually does not depend on the condition (H_0 or H_1).

The ratio of likelihoods is then

$$\frac{L(x_1,\ldots,x_m,\mu_1)}{L(x_1,\ldots,x_m,\mu_0)} = e^{-\frac{1}{2\hat{\sigma}^2_{MLE}}\sum_{i=1}^{m}(x_i-\mu_1)^2-(x_i-\mu_0)^2}. \tag{6.4}$$

The iterations and bounds are equivalent to section 5.3.

Practically, we notice that:

$$\begin{aligned}\frac{L(x_1,\ldots,x_m,\mu_1)}{L(x_1,\ldots,x_m,\mu_0)} &= e^{-\frac{1}{2\hat{\sigma}^2_{MLE}}\sum_{i=1}^{m}(x_i-\mu_1)^2-(x_i-\mu_0)^2} \\ &= e^{-\frac{\mu_0-\mu_1}{2\hat{\sigma}^2_{MLE}}\sum_{i=1}^{m}(2x_i-\mu_1-\mu_0)} \\ &= e^{-\frac{1}{2\hat{\sigma}^2_{MLE}}\left(2(\mu_0-\mu_1)(\sum_{i=1}^{m}x_i)+m\mu_1^2-m\mu_0^2\right)},\end{aligned}$$

with

$$\begin{aligned}\hat{\sigma}^2_{MLE} &= \frac{1}{m}\sum_{i=1}^{m}\left(X_i-\frac{1}{m}\sum_{i=1}^{m}X_i\right)^2 \\ &= \frac{1}{m}\left(\sum_{i=1}^{m}X_i^2-\frac{1}{m}\left(\sum_{i=1}^{m}X_i\right)^2\right),\end{aligned}$$

so we just store $\sum_{i=1}^{m} x_i$ and $\sum_{i=1}^{m} x_i^2$ in the sampler to compute the likelihood ratio.

6.4 SPRT experiments

The figure 6.3 presents the experimentation of this technique with the *cluster.sm* and *dice.pm* examples. The number of iterations required by SPRT depends on θ, so these numbers are displayed in figure 6.4 in terms of θ.

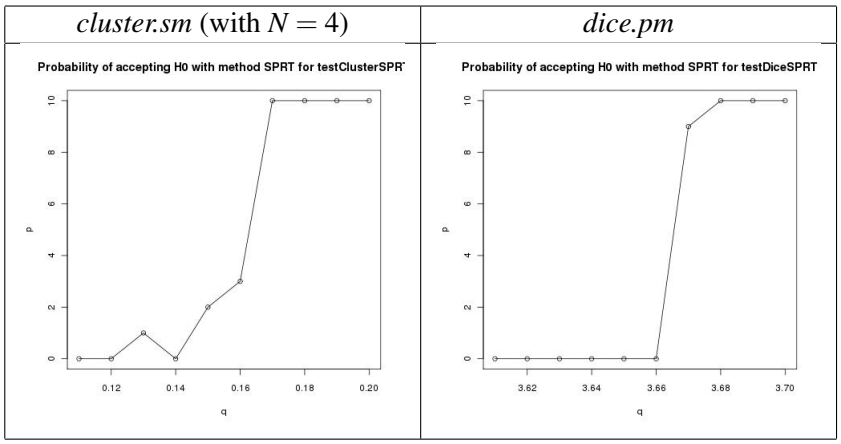


Figure 6.3: Comparisons of the probabilities of accepting H_0 hypothesis (ordinates) with Bartlett's SPRT, for $\alpha_e = 0.1$, $\beta_e = 0.1$, $\delta' = 10^{-2}$, and 10 runs of PRISM per value of θ (abscissa).

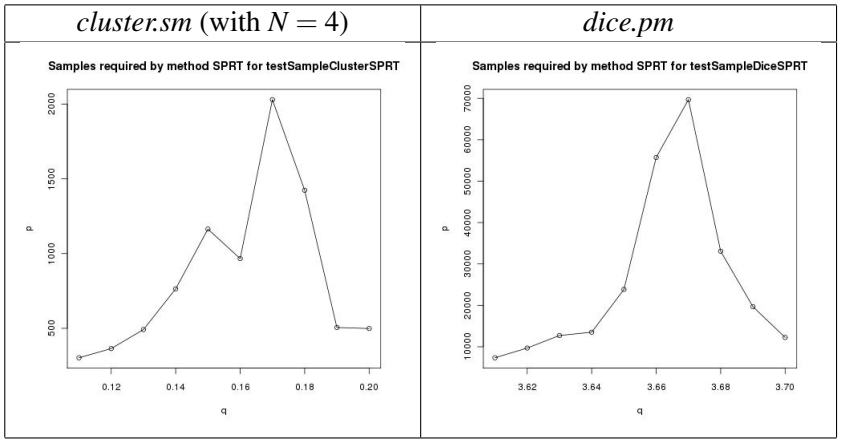


Figure 6.4: Average number of iterations required by SPRT in terms of θ. SPRT is run with $\alpha_e = 0.1$, $\beta_e = 0.1$, $\delta' = 10^{-2}$, and 10 runs are used to compute each point of the graph.

6.5 Summary

The chapter extended the test methods to rewards computation. The first way of checking a property with threshold reward is unchanged: we compute the estimation, and compare it with the threshold reward, given the parameters. It remains optimal using CI method, with respect to the parameters.

The Wald's sequential test does not extend to the normal distribution the rewards are following. We use Bartlett's sequential test instead, which estimates the nuisance parameter, the variance, by the maximum likelihood estimator for this distribution. We validate this experimentally with the examples we used previously.

These methods have been implemented in the console mode of PRISM (to use them, read appendix A).

Chapter 7

Conclusion

The initial objective of the project was to enhance PRISM with good statistical model checking support. Statistical probabilistic model checking can be divided in several components (see figure 2.5): generation, verification and analysis. I chose to improve the last part, which was limited in PRISM to an implementation of the AMC method only for probabilities estimation.

The literature offers often different presentations of the statistical model checking analysis for probabilities, with sometimes some empirical explanations. In this dissertation, these methods, regrouped into CI, ACI and AMC, have been statistically explained, tested on real-size biological examples - which require statistical model checking because of the number of states - and compared.

The methods have also been extended to the context of reward expectation. If it is natural for CI and ACI methods, which rely on the central-limit theorem, this required additional assumptions and theorems to end up with methods comparable to AMC. Bartlett's sequential probability ratio test has been proposed to substitute the Wald's one, no more valid in the case of rewards, because rewards random variables are following normal distribution.

Globally, one cannot conclude that a method is better in absolute than the others. CI and SPRT are optimal in terms of sampling, AMC knows the required number of samples before the simulation, ACI is certainly faster but unsure in terms of reliability, *etc*. The different parameters the methods take have also their own meanings, and it can be more convenient in some situations to use type I and II error rather than level of confidence, or the opposite. These methods also have specific assumptions, which can be accepted or not in certain applications. To help the user[1] in choosing the most convenient method for his problem, I propose the

[1] Especially the PRISM user!

following table summing up all the methods presented in this dissertation, with the problems they solve, the parameters they require, the conditions they must respect, some comments about their efficiency observed on the examples and some recommendations about when to use (or not to use) them.

The initial objectives of this thesis have been fulfilled: the modified version of PRISM I worked on provides a good support for reliable statistical estimations of probabilities and rewards. It offers several methods for each problem, and some decisions can be automatically made in terms of the parameters. The extensions to rewards computation I developed are efficient for CI and ACI, but a bit disappointing for AMC: the two AMC extensions I proposed seem to be really inefficient (bounded expectation) or not reliable in some cases (bounded variables, when some paths have a reward bigger than the given bound - see limit test 4.3.1)[2]. On the basis of the observations, a table is finally proposed to help in the selection of the most adapted method to a context.

I had started to study the case of statistical model checking for *Markov Decision Process*, but this question is very vast and still open, and could certainly constitute the subject of a thesis. Some improvements could also certainly be realized in the first component of the PRISM statistical model checker, the generation of path, using some specific optimizations of Monte Carlo method. A last area which has not been considered here but could be investigated is the approach of model checking with Bayesian statistics (read [14, 35]).

[2]The next version of APMC will provide a similar support for rewards.

Problem:	Method:	When to use it:	Parameters:	Conditions on parameters:	Efficiency:
$P_{=?}[\phi]$	CI	reliable, unknown number of iterations	2 among $\{\alpha, w, n\}$	none	optimal
	ACI	approximative, unknown number of iterations	2 among $\{\alpha, w, n\}$	none	can be faster than CI
	AMC	reliable, parameters computable before simulation	2 among $\{\delta, \varepsilon, n\}$	$n\varepsilon^2 \geq \frac{ln2}{2}$	compute more iterations than required
$P_{\bowtie\theta}[\phi]$	CI	reliable, unknown number of iterations	2 among $\{\alpha, w, n\}$, θ	none	optimal, but can require new simulations with refined parameters
	ACI	approximative, unknown number of iterations	2 among $\{\alpha, w, n\}$, θ	none	faster than CI, can require new simulations with refined parameters
	AMC	reliable, parameters computable before simulation	2 among $\{\delta, \varepsilon, n\}$, θ	$n\varepsilon^2 \geq \frac{ln2}{2}$	compute more iterations than required, can require new simulations with refined parameters
	Wald's SPRT	reliable, unknown number of iterations	θ, δ', α_e, β_e	none	optimal [33]

Problem:	Method:	When to use it:	Parameters:	Conditions on parameters:	Efficiency:
$R_{=?}[\phi]$	CI	reliable, unknown number of iterations	2 among $\{\alpha, w, n\}$	none	optimal
	ACI	approximative, unknown number of iterations	2 among $\{\alpha, w, n\}$	none	can be faster than CI
	bounded extended AMC	reliable if a X_i bound *maxReward* known, parameters computable before simulation, to be used when variance not abnormal (see figure 4.12)	2 among $\{\delta, \varepsilon, n\}$, *maxReward*	$n\varepsilon^2 \geq \frac{maxReward \times ln2}{2}$, $max_i\{X_i\} \leq maxReward$	compute (really) more iterations than required, depend on the precision of *maxReward*
	unbounded extended AMC	reliable if a $E[\chi_{P,\Sigma}]$ bound γ known, parameters computable before simulation, to be used in case of very high variance (see figure 4.12)	2 among $\{\delta, \varepsilon, n\}$, γ	$n\varepsilon^2 \geq \frac{(L_{min}+\gamma)^2 ln\left(\frac{2L}{L_{min}-\gamma}\right)}{2}$, $E[\chi_{P,\Sigma}] \leq \gamma$	compute (really) more iterations than required, depend on the precision of γ

Problem:	Method:	When to use it:	Parameters:	Conditions on parameters:	Efficiency:
$R_{\bowtie\theta}[\phi]$	CI	reliable, unknown number of iterations	2 among $\{\alpha, w, n\}$, θ	none	optimal, but can require new simulations with refined parameters
	ACI	approximative, unknown number of iterations	2 among $\{\alpha, w, n\}$, θ	none	can be faster than CI, can require new simulations with refined parameters
	bounded extended AMC	reliable if a X_i bound *maxReward* known, parameters computable before simulation, to be used when variance not abnormal (see figure 4.12)	2 among $\{\delta, \varepsilon, n\}$, *maxReward*, θ	$n\varepsilon^2 \geq \frac{maxReward \times ln2}{2}$, $max_i\{X_i\} \leq maxReward$	compute (really) more iterations than required, depend on the precision of *maxReward*; can require new simulations with refined parameters
	unbounded extended AMC	reliable if a $E[\chi_{P,\Sigma}]$ bound γ known, parameters computable before simulation, to be used in case of very high variance (see figure 4.12)	2 among $\{\delta, \varepsilon, n\}$, γ, θ	$n\varepsilon^2 \geq \frac{(L_{min}+\gamma)^2 ln\left(\frac{2L}{L_{min}-\gamma}\right)}{2}$, $E[\chi_{P,\Sigma}] \leq \gamma$	compute (really) more iterations than required, depend on the precision of γ; can require new simulations with refined parameters
	Bartlett's SPRT	reliable, unknown number of iterations	θ, δ', α_e, β_e	none	optimal

Tables legend:

α	level of confidence
w	half-width of the confidence interval
n	number of required iterations
δ	confidence parameter
ε	approximation parameter
θ	threshold probability or reward
δ'	value to make the two hypotheses independent
α_e	type I error
β_e	type II error
$maxReward$	maximum reward
γ	an upper bound of the expectation
$E[\chi_{P,\Sigma}]$	reward expectation
$max_i\{X_i\}$	maximum reward in the paths sample
L_{min}	value minimizing the function $f(x) = \frac{(x+\gamma)^2}{2\varepsilon^2} ln\left(\frac{2x}{\delta x-\gamma}\right)$
CI	confidence interval method
ACI	asymptotic confidence interval method
AMC	approximate model checking method
$SPRT$	sequential probability ratio test

Bibliography

[1] C. Baier, J.-P. Katoen, *Principles of Model Checking*, MIT Press, Cambridge, 2007.

[2] N. Barkai, S. Leibler, *Biological rhythms: Circadian clocks limited by noise*, Nature, vol. 403, pp. 267-268, 2000.

[3] V. Bentkus, *An Extension of the Hoeffding Inequality to Unbounded Random Variables*, Lithuanian Mathematical Journal, vol. 48, n. 2, pp. 137-157, Springer, 2008.

[4] A. J. Bertie, *Java Statistical Classes Library*, 2005. http://www.jsc.nildram.co.uk/

[5] F. Didier, T.A. Henzinger, M. Mateescu, V. Wolf, *Approximation of Event Probabilities in Noisy Cellular Processes*, CMSB 2009, LNBI 5688, pp. 173-188, Springer-Verlag, Berlin Heidelberg 2009.

[6] M. Duflot, M. Kwiatkowska, G. Norman, D. Parker, *A formal analysis of Bluetooth device discovery*, International Journal on Software Tools for Technology Transfer (STTT), vol. 8, n. 6, pp. 621-632, October 2006.

[7] P.I. Good, J.W. Hardin, *Common Errors in Statistics (and how to avoid them)*, third edition, Wiley, 2009.

[8] B. Haverkort, H. Hermanns, J.-P. Katoen, *On the use of model checking techniques for dependability evaluation*, Proc. 19th IEEE Symposium on Reliable Distributed Systems, pp. 228-237, 2000.

[9] J. Health, M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, *Probabilistic model checking of complex biological pathways*, Proc. Computational Methods in Systems Biology, Lecture Notes in Bioinformatics, vol. 4210, pp. 32-47, Springer Verlag, 2006.

[10] J. Health, M. Kwiatkowska, G. Norman, D. Parker, O. Tymchyshyn, *Probabilistic model checking of complex biological pathways*, Theoretical Computer Science, vol. 319, n. 3, pp. 239-257, 2008.

[11] T. Hérault, R. Lassaigne, F. Magniette, S. Peyronnet, *Approximate Probabilistic Model Checking*, VMCAI 2004, LNCS 2937, pp. 73-84, Springer-Verlag, Berlin Heidelberg 2004.

[12] W. Hoeffding, *Probability Inequalities for Sums of Bounded Random Variables*, Journal of the American Statistics Association, vol. 58, Mars, n. 301, pp. 13-30, 1963.

[13] O. Ibe, K. Trivedi, *Stochastic Petri Net Models of Polling Systems*, In IEEE Journal on Selected Areas in Communications, Vol. 8, n. 9, pp. 1649-1657, 1990.

[14] S.K. Jha, E.M. Clarke, C.J. Langmead, A. Legay, A. Platzer, P. Zuliani, *A Bayesian Approach to Model Checking Biological Systems*, CMSB 2009, LNBI 5688, pp. 218-234, Springer-Verlag, Berlin Heidelberg 2009.

[15] D. E. Knuth, A. C. Yao, *The complexity of nonuniform random number generation*, in *Algorithm and Complexity*, Academic Press, New York 1976.

[16] L.L. Kupper, K.B. Hafner, *How Appropriate are popular Sample Size Formulas?*, in *The American Statistician*, vol. 43, May, n. 2, pp. 101-105, 1989.

[17] M. Kwiatkowska, G. Norman, D. Parker, *Stochastic Model Checking*, LNCS 4486, pp. 220-270, Springer-Verlag, 2007.

[18] R. Lassaigne, S. Peyronnet, *Probabilistic Verification and Approximation*, Annals of Pure and Applied Logic, n. 152, pp. 122-131, Elsevier, 2007.

[19] A. Law, D. Kelton, *Simulation Modelling and Analysis*, McGraw-Hill Education, New York 2000.

[20] P. Lecca, C. Priami, *Cell cycle control in eukaryotes: A BioSpi model*, Proc. Workshop on Concurrent Models in Molecular Biology, ENTCS, 2003.

[21] J. Li, D. Jeske, *Bartlett and Wald Sequential Hypothesis Testing with Correlated Data*, Quality & Productivity Research Conference, New York 2009. http://www.amstat-online.org/sections/qp/qprc/2009/papers/QPRC_Contributed_Session_4/Judy_Bartlett_QPRC09.pdf

[22] P. Massart, *Concentration Inequalities and Model Selection*, Lecture Notes in Mathematics, n. 1896, Springer, 2003.

[23] D. Parker, *Probabilistic Model Checking*, lectures, Oxford University, 2009. http://www.prismmodelchecker.org/lectures/pmc/

[24] A. Pnueli, L. Zuck, *Verification of Multiprocess Probabilistic Protocols*, Proceedings of the third annual ACM symposium on Principles of distributed computing, p.12-27, August 27-29, Vancouver 1984.

[25] S.M. Ross, *Simulation*, third edition, Academic Press, 2002.

[26] W. Sandmann, C. Maier, *On the Statistical Accuracy of Stochastic Simulation Algorithms implemented in Dizzy*, Proc. WCSB, pp. 153–156, 2008.

[27] K. Sen, M. Viswanathan, G. Agha, *On Statistical Model Checking of Stochastic Systems*, CAV 2005, LNCS 3576, pp. 266-280, Springer-Verlag, Berlin Heidelberg 2005.

[28] K. Sen, M. Viswanathan, G. Agha, *Statistical Model Checking of Black-Box Probabilistic Systems*, R. Alur, D.A. Peled (Eds.), Proceedings of the 16th International Conference on Computer Aided Verification, pp. 202-215, Springer, Berlin 2004.

[29] P. K. Shah, D. R. Jeske, R. F. Luck, *Sequential Hypothesis Testing Techniques for Pest Count Models With Nuisance Parameters*, Journal of Economic Entomoly, vol. 102, n. 5, pp. 1970-1976, 2009.

[30] T. Tao, *Concentration of Measure Notes*, 2010. http://terrytao.wordpress.com/2010/01/030254a_notes_1_contration_of_measure/

[31] J. Vilar, H.-Y. Kueh, N. Barkai, S. Leiber, *Mechanisms of noise-resistance in genetic oscillators*, Proc. Nat Acad Sci, vol. 99, n. 9, pp. 5988-5992, USA 2002.

[32] G. Barrie Wetherill, *Sequential Methods in Statistics*, chap. 4, Wiley, 1966.

[33] H.L.S. Younes, M. Kwiatkowska, G. Norman, D. Parker, *Numerical vs. Statistical Probabilistic Model Checking*, International Journal on Software Tools for Technology Transfer (STTT), vol. 8, June, n. 3, pp. 216-228, Springer, 2006.

[34] D. Zhang, Z. Wang, *Probability Inequalities for Sums of Independent Unbounded Random Variables*, Applied Mathematics and Mechanics, vol. 22, May, n. 5, Shanghai 2001.

[35] P. Zuliani, E.M. Clarke, A. Platzer, *Bayesian Statistical Model Checking with Application to Stateflow/Simulink Verification*, HSCC, April 12-15, Stockholm 2010.

[36] *Colt Library*, CERN, 1999. http://acs.lbl.gov/software/colt/

Appendix A

User guide for Statistical Model Checking with enhanced PRISM

A good support for statistical approaches of probabilistic model checking has been added to a development version of PRISM (present on the attached disk), and will be integrated soon to the distributed version[1]. This provides all the methods described in the thesis. Because there is lots of possible parameters, options, automatic decisions and so on, these few pages offer a guide for the user who wants to use PRISM statistically.

A.1 Graphic mode

The graphic mode of PRISM with statistics is easy to use, but not convenient for automation of computation (see appendix B). It implements all the methods from chapters 3 and 4, except the unbounded AMC method. To get information about the console mode, which implements all the methods, read section A.2.

PRISM GUI offers the possibility of estimating the probability or reward of one property[2]. To launch a simulation, add a model, create or add a property, and click on this property and select *simulation*. A dialogue box is displayed, with several fields (figure A.1). In the first one, one method among CI, ACI and AMC can be chosen. Then select the parameter which is not supposed to be known. If the CI or ACI method has been selected, this parameter will be computed after the simulation. If AMC is used, the missing parameter is computed continuously. The last field defines the maximum length of each path picked from the model.

[1] http://www.prismmodelchecker.org/

[2] Console mode can test several properties a time - read section A.2.

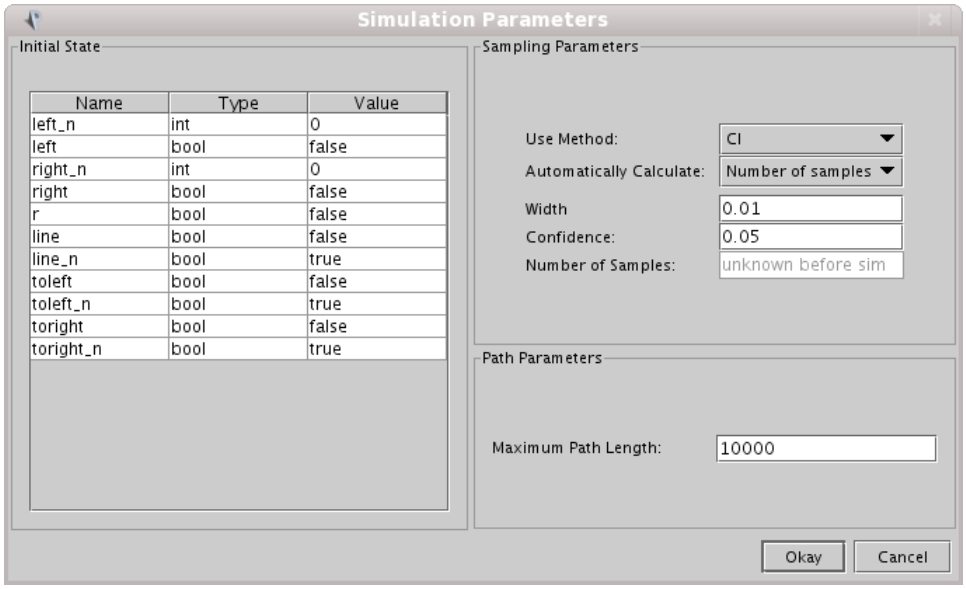


Figure A.1: Dialogue box of method selection in graphic mode

Note that if AMC method has been selected with the confidence parameter δ as missing parameter, if the approximation parameter ε and the number of iterations N are such that $N\varepsilon^2 < \frac{ln\left(\frac{2}{\delta}\right)}{2}$, or $N\varepsilon^2 < \frac{maxReward \times ln\left(\frac{2}{\delta}\right)}{2}$ if it is a reward property, the last field will ask to either decrease the accuracy or increase the number of iterations.

If CI or ACI method is used, the decision concerning the possible null variance will be made on the base of inequalities 3.6 and 4.1. To impose manually a minimum number of iterations before claiming whether it is a null variance or not, just uncheck the box *automatic decision* in the dialogue box (figure A.2) of *options*, *simulation*, and specify the number of iterations in the *number of iterations to conclude* field.

For reward properties, the *maxReward* for AMC method can be defined in the dialogue box (figure A.2) of *options*, *simulation*.

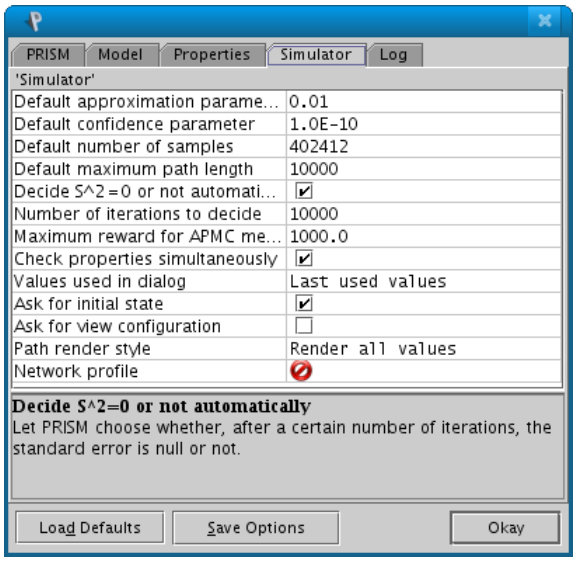


Figure A.2: Dialogue box of simulation options in graphic mode

A.2 Console mode

The console mode implements all the methods presented in the dissertation, and can be easily automatized using Unix/Dos environment.

To launch the model checking of a property *prop* over a model *model*, the command[3] is

```
./prism model -csl prop
# numerical model checking of prop on model
```

Now to use statistical model checking instead of the classic numerical approach, add *-sim* and define two parameters among *-simapprox*[4], *-simconf*[5] and *-simsamples* - the number of iterations.

```
./prism model -csl prop -sim -simapprox 0.01 -simconf
\ 0.05
# statistical model checking of prop on model with AMC
# method, with epsilon=0.01 and delta=0.05
```

[3] *-csl* or *-pctl*

[4] Approximation parameter if AMC, width of the CI if CI or ACI.

[5] Confidence parameter for AMC, level of confidence for CI and ACI.

If three parameters are provided, the confidence parameter is ignored. If only one or no parameter is given, the default values are considered.

The default selected method is AMC. To use CI or ACI instead, just add respectively *-CI* and *-ACI*.

```
./prism model -csl prop -sim -simapprox 0.01 -simconf
\ 0.05 -CI
# statistical model checking of prop on model with CI
# method, with width=0.01 and alpha=0.05
```

The maximum depth of each path in the model can be defined with *-simpathlen*:

```
./prism model -csl prop -sim -simapprox 0.01 -simconf
\ 0.05 -CI -simpathlen 10000
```

Until now, the decision concerning the nullity of the variance was made automatically by equations 3.6 and 4.1. If we want to specify it manually, for a limit case for instance, then we add the option *-simmanual*. The number of required iterations to conclude is then the default one - to use another one, add *-simvar*:

```
./prism model -csl prop -sim -simapprox 0.01 -simconf
\ 0.05 -CI -simmanual -simvar 1000
# statistical model checking of prop on model with CI
# method, with width=0.01 and alpha=0.05. The decision
# is made after 1000 iterations.
```

If we consider a reward property, the default AMC method is the AMC with bounded variables. A default maximum reward value is taken into account, but it can be modified with the option *-simmaxrwd*:

```
./prism model -csl prop.rwd -sim -simapprox 0.01 -simconf
\ 0.05 -simmaxrwd 1000
# statistical model checking of prop on model with AMC
# method with bounded variables (bound: 1000), with
# width=0.01 and alpha=0.05.
```

To use the unbounded AMC method (or method with bounded expectation), add *-ext* to the command line. The expectation's bound takes a default value, which can be specified with *-simmaxrwd*.

```
./prism model -csl prop.rwd -sim -simapprox 0.01 -simconf
\ 0.05 -ext -simmaxrwd 10
# statistical model checking of prop on model with
# unbounded AMC method, with width=0.01 and alpha=0.05.
```

All these options can be used simultaneously.

Finally, to use the SPRT method, just specify the type I error α, the type II error β and the δ' after the option *-sprt*.

```
./prism model -csl prop -sim -sprt 0.1 0.1 0.1
# statistical model checking of prop on model with
# SPRT method with alpha=0.1, beta=0.1, delta'=0.1.
```

Appendix B

Strategies of automation for simulations

Once the different methods have been implemented, it is interesting to test them on real models and check experimentally that their behaviour is consistent. If the width/approximation parameter and the number of iterations can be checked directly with one run of PRISM, it is not the case of the confidence parameter/level of confidence (see subsection 3.5.1), and we have to develop some strategies to automatize the simulations, the collection of the data and their treatment and display.

We use the Linux environment, with our modified version of PRISM, to get the results, GREP and SED to collect and format the data, BC to compute reals in the Unix environment, and R to sketch graphs from data. A BASH script automatizes all these operations. Some examples of this script can be found on the attached disk.

All the strategies for probabilities and rewards estimations follow the following scheme:

- compute numerically the probability/reward with PRISM (if relevant[1])
- for each method,
  - compute statistically the probability/reward with PRISM
  - get and store the result, with GREP and SED
- generate a R command file with the data and the parameters[2]

[1] This is not possible for the biological examples, which are out of the range of the numerical resolution of PRISM.

[2] To get the area of confidence in R, we need the quantile $1 - \frac{\alpha}{2}$ - we use BC to compute it.

- make R generate a graph of the distribution of the estimations, with the area of confidence for the confidence parameter.

This generates graphs such as figure 3.4.

To verify the properties with bounded probabilities/rewards, we check the acceptance regions of the test. The graphs like in figure 5.2 are generated with scripts which have the following scheme:

- specify the threshold probability/reward range to test $[\theta_{min}; \theta_{max}]$, and the number of steps N
- specify the number S of samples per step
- for each step
  - compute the corresponding threshold probability/reward using BC
  - for S iterations
    - compute the result using PRISM
    - get the result with GREP and SED
    - update the average of the results
  - write in a R command file the pair (computed threshold, average)
- make R generate a graph linking the points.