Simulation + Hypothesis Testing for Solving the Probabilistic Model Checking Problem

Axel Legay and Mahesh Viswanathan

September 11, 2009

Contents

2 Introduction

Objective

Our Objectives

We will have four objectives :

1. Getting more knowledge about how to verify stochastic systems;
2. Studying how statistical techniques can be applied in this area;
3. New applications (models, properties);
4. Going further than algebraic approaches (PRISM, LIQOR, ...).

Stochastic Systems

Stochastic Systems (1)

Definition 1. A stochastic system is a process that evolves over time, and whose evolution can be predicted in terms of probability (pure) and nondeterministic choices (nonpure).

Where can they be found?

- Embedded systems;
- Economy;
- Networking;
- Systems Biology;
- ...

Stochastic Systems (2) : Models

Those we consider

- Any model of pure stochastic systems;
- Example : Markov Chains.

Those we won't consider

- Any model which mixes nondeterministic and probabilistic choices;
- Example : Markov Decision Processes (MDPs).

Verifying Stochastic Systems

Verification Process

Question

$$\text{Does } \mathcal{S} \models P_{\geq\theta}(\phi) \text{ ?}$$

where :

- $\mathcal{S}$ is a Stochastic system;
- Executions of $\mathcal{S}$ are sequences of states (random variables);
- ϕ is some execution-based property (specification language);
- $P(X)$ means : "the probability for X to happen";
- θ is a probability threshold.

3 The Algebraic Approach

Outline

Contents

Description of the approach

Main idea

Overview

- Assume the existence of a probability space;
- Compute the probability p for $\mathcal{S}$ to satisfy ϕ;
- Compare p with θ.

Difficulty

Algorithms to compute p.

Advantages and Disadvantages

Advantages

- High accuracy in result;
- Exists for nonpure models such as MDPs;
- Well-established tools : PRISM, LIQUOR, PMAUD, ...
- ...

Disadvantages
There are at least 5 disadvantages

1. Memory intensive;
2. Limited to certain classes of systems and properties (finite-state, ..);
3. No unique solution;

4. Complex algorithms :

   One Difficulty: How to find efficient data structures;

5. Difficult to parallelize.

4 Statistical Model Checking

Outline

Contents

Description of the approach

Learning from a Simple Problem

A (VERY) simple problem

- Consider a machine that flips a (possibly biaised) coin;
- Is the probability p of having a head greater or equal to some θ?

A solution

- Do several flips and deduce the answer from them;
- If the number of flips is infinite, our answer will be correct up to some type error.

This is the statistical model checking approach!

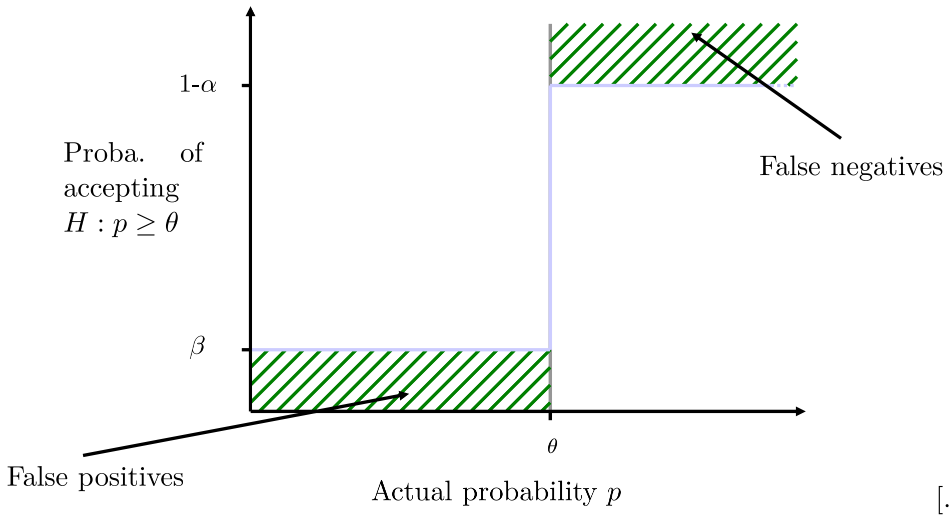
Hypothesis Testing

$$\text{Test } P(\text{having a head}) \geq \theta \text{ against } P(\text{having a head}) < \theta$$

With (Type error):

1. α : the probability to accept H_1 while H_0 is true;
2. β : the probability to accept H_0 while H_1 is true.

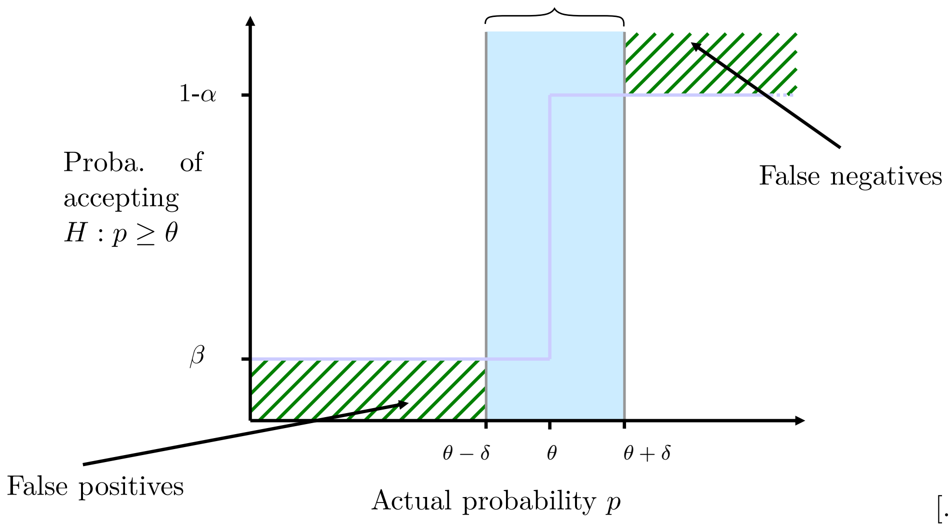
Performance of Test



[.5cm]

Needs an infinite number of samples to get ideal performances !

Performance of Test



[.5cm]

If $p \in [\theta - \delta, \theta + \delta]$, we say we are *indifferent* to know if $p \geq \theta$

Summary

We want to test :

$H_0 : p \geq p_0$ against $H_1 : p < p_1$, where $p_0 = \theta + \delta$ and $p_1 = \theta - \delta$.

With:

- Type erros α and β, and
- Indifference region 2δ.

Bernouili Variables for experiments

Bernouili variable X_i of parameter p

- Takes two values : $X_i = 0$ or $X_i = 1$;
- $P[X_i = 1] = p$ and $P[X_i = 0] = 1 - p$;
- Realization is denoted x_i.

Experiments

- We assume independent trials;
- We can generate as much trials as we want;
- p is the probability to get a head ;
- Associate a bernouili variable X_i to each trial;
- $X_i = 1$ iff the trial is a tail.

Two Algorithms

Algorithm 1 : Single Sampling plan

- Pre-compute a number n of experiments;
- n depends on δ,α, and β.

Algorithm 2
Basically a on-the-fly version of the Single Sampling Plan (in fact, this is much more :-)!)

Single Sampling plan

Single Sampling plan : Principles

- Choose n and c with $c \leq n$;
- n observations $x_1, \ldots, x_n$ for n samplings $X_1, \ldots, X_n$;
- $Y = \sum_{i=1}^{n} x_i$;
- Accept H_0 if $Y \geq c$ and H_1 otherwise;

Difficulty : Find n and c such that α and β are satisfied

Single Sampling plan : α and β

Definition 2. $P[Y \leq c] = F(c; n; p) = \sum_{i=0}^{i=c} C_i^n p^i (1-p)^{n-i}$.

Definition 3. $F(c; n; \theta)$: probability to accept H_1.

Definition 4.
- $F(c; n; \theta + \delta) \leq \alpha$;
- 1 - $F(c, n; \theta - \delta) \leq \beta$.

Single Sampling plan : Disadvantages

- Difficult to find c and n : *No unique solution*;
- Difficult to minimize n;

Approximation algorithms exist (Haakan Youness).

Better for black-box systems (next part of the tutorial)..

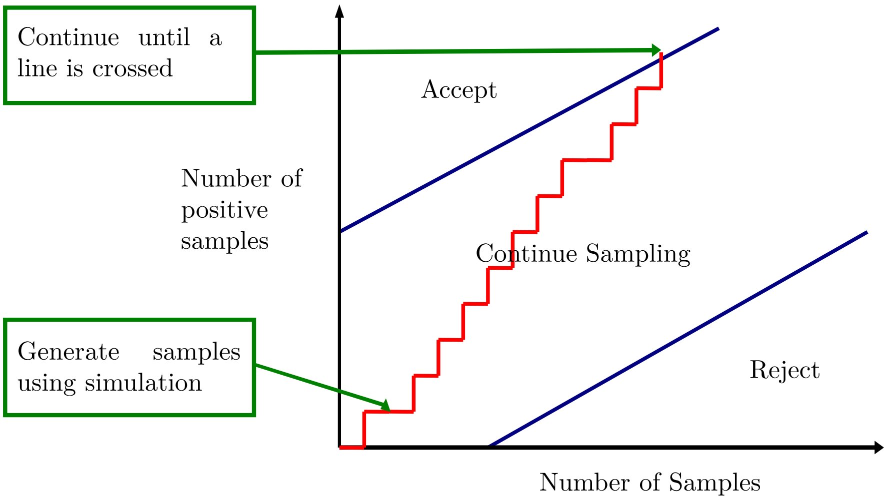
Optimality (Hasting)

Thresholds		Values	
$\theta - \delta = 0$	$\theta + \delta = 1$	$n = 1$	$c = 0$
$\theta - \delta = 0$	$\theta + \delta < 1$	$n = \frac{\log \alpha}{\log 1 - \theta - \delta}$	$c = 0$
$\theta - \delta > 0$	$\theta + \delta = 1$	$n = \frac{\log \beta}{\log \theta - \delta}$	$c = n - 1$

Sequential Hypothesis Testing

Sequential Hypothesis Testing

- Check hypothesis after each sample and stop as soon as possible
- We can find an *acceptance line* and a *rejection line* given $\alpha, \beta, \theta,\ \delta$.[.6cm]



Wald's Testing

Compute

$$W = \prod_{i=1}^{m} \frac{Pr(X_i = x_i \mid p = \theta - \delta)}{Pr(X_i = x_i \mid p = \theta + \delta)} = \frac{(\theta-\delta)^{d_m}(1-\theta+\delta)^{m-d_m}}{(\theta+\delta)^{d_m}(1-\theta-\delta)^{m-d_m}}, \quad (1)$$

where $d_m = \sum_{i=1}^{m} x_i$. Stop when :

- $W \geq (1-\beta)/\alpha$: H_1 is accepted;
- $W \leq \beta/(1-\alpha)$: H_0 is accepted.

More Mathematics

- *In theory :* the test does not guarantee α and β!
- New parameters α' and β' such that
  - $\alpha' \leq \frac{\alpha}{1-\beta}$ and $\beta' \leq \frac{\beta}{1-\alpha}$

- $\alpha' + \beta' \leq \alpha + \beta$;

- *In practice :* one observes that α and β are almost often guarantee, and it may even be better!

Example 5. Let $p_0 = 0.5$, $p_1 = 0.3$, $\alpha = 0.2$, $\beta = 0.1$:

- In theory : $\alpha' \leq \frac{0.2}{0.9} = 0.222...$ and $\beta' \leq \frac{0.1}{0.8} = 0.125$;
- Computer simulation : $\alpha' = 0.175$ and $\beta' = 0.082$.

Performances (1)

- Single sampling plan can be better than SPRT !
- SPRT is, in practice, more efficient;
- Expected sample size E_p (Wald's formula) :
  - SPRT minimizes E_p at $\theta + \delta$ and $\theta - \delta$;
  - E_p increases from 0 to $\theta - \delta$;
  - E_p decreases from $\theta + \delta$ to 1;
  - Between $\theta - \delta$ and $\theta + \delta$: increase and then decrease.

Performances (2) : SPRT

Indifference region 2δ	Number of executions
0.1	55
0.05	106
0.02	228
0.01	627
0.005	1056

Number of trajectories against 2δ

$(\alpha = \beta = 0.02)$

- m increases linearly if δ decreases.

Performances (3) : SPRT

Test strength $\alpha(=\beta)$	Number of executions
$1e^{-2}$	335
$1e^{-4}$	502
$1e^{-6}$	857
$1e^{-8}$	1301
$1e^{-10}$	1467

Number of trajectories against α ($\beta = \alpha$ and $2\delta = 0.02$)

- m increases logarithmically if α and/or β decrease.

From Flipping a coin to Model Checking

Fact
Flipping a coin is nothing more than testing whether a finite execution satisfies a property.

Results
Consequence : Wald's testing directly applies to model check properties of white-box stochastic systems.

Properties

- *Natural :* those that can be checked on finite execution;
- *Going further :* Some properties on infinite executions:

Next part of the tutorial.

Why are nonpure systems forbidden?

- We sample a unique distribution;
- Sampling several distributions would require to distinguish between them;
- This cannot be done on the sole basis of running the system.

Advantages

Advantages

- Easy to parallelize (independent sampling, unbiased distributed sampling);
- Independent of system's size;
- Independent of system's probability distribution;
- Easy to trade accuracy for speed;
- Uniform approach;
- Easy to implement :
  - In most cases, one only need to implement a "trace checker" that tests whether an execution satisfies a given property;
  - No need for complex data structures.

A Note on Parallelization

- Observations are generated by different machines;
- Observations must be independent; :
  - Using different seeds is not sufficient : it only determines initial numbers, not the way the sequence is generated;
  - Solution :

    encode process ID directly is the generator.
- Slave - master : experiments are collected in ring-order.

5 Experiments

2 types of experiments

- $\Delta - \Sigma$ Modulator (conversion : analogue to digital);
- Systems biology (briefly).

Modulator

Model Checking mixed-signal circuits

- Mature for digital designs but still new for analog and mixed design
- Difficult due to continuous and hybrid state variables

Probabilistic Model Checking

- Stochastic systems and/or stochastic uncertainties
- Exact solution is a difficult problem in general

Statistical Approach

- Use of numerical simulation
- Approximate solution with bounds on errors

Systems and Logics with Signals

Outline

Contents

Logics: LTL formulas

Let $\mathcal{B}$ be a set of predicates. The following defines an LTL formula:

$$\phi ::= \mathbf{T} \,|\, \mathbf{F} \,|\, b \in \mathcal{B} \,|\, \neg\phi \,|\, \phi_1 \vee \phi_2 \,|\, \bigcirc \phi \,|\, \phi_1 \mathcal{U} \phi_{\in}.$$

Let $\omega = s_1 s_2 ... s_k$, $|\omega| = k$, $\omega^i = s_i s_{i+1} ... s_k$, $\omega(i) = s_i$ and L be a mapping from S to $2^{\mathcal{B}}$. We have:

- $\omega \models \mathbf{T}$, $\omega \not\models \mathbf{F}$ and $\omega \models \neg\phi$ iff $\omega \not\models \phi$
- $\omega \models b$ with $b \in \mathcal{B}$ iff $b \in L(\omega(0))$
- $\omega \models \phi_1 \vee \phi_2$ iff $\omega \models \phi_1$ or $\omega \models \phi_2$
- $\omega \models \bigcirc\phi$ iff $|\omega| > 1$ and $\omega^1 \models \phi$
- $\omega \models \phi_1 \mathcal{U} \phi_{\in}$ iff there exists $0 \leq i \leq |\omega| - 1$ such that $\omega^i \models \phi_2$, and for each $0 \leq j < i$, $\omega^j \models \phi_1$

Additionally, we use the *eventually* operator $\Diamond$ defined as $\Diamond\phi = \mathbf{F}\mathcal{U}\phi$. Note that we only consider *finite* executions.

Logics: Execution Predicates

Definition 6 (Execution Predicate)**.** Let $\Sigma(\mathcal{S})$ be the set of all the executions of an SSDES $\mathcal{S}$. An *execution predicate* p for $\mathcal{S}$ is a mapping $p : \sigma \in \Sigma(S) \mapsto p(\sigma) \in \{\mathbf{T}, \mathbf{F}\}$.

Example 7. Execution predicate p that decides whether the mean value of the analog signal associated with σ is ≥ 0:

$$p(\sigma) = \mathbf{T} \quad \text{iff} \quad \frac{1}{N}\sum_{k=0}^{N-1} \pi_a(\sigma(k)) \geq 0.$$

More complex functionals such as the Fourier transform can be used[.4cm]

Claim. *Let $\mathcal{S}$ be an SSDES and ϕ be a Boolean combination of LTL formulas and execution predicates. One can always associate a probability with the set of executions of $\mathcal{S}$ that satisfy ϕ.*

A Class of Mixed-Signal Circuits: $\Delta - \Sigma$ Modulators

Outline

Contents

$\Delta - \Sigma$ Modulators for Dummies

Analog to Digital converters (ADC)

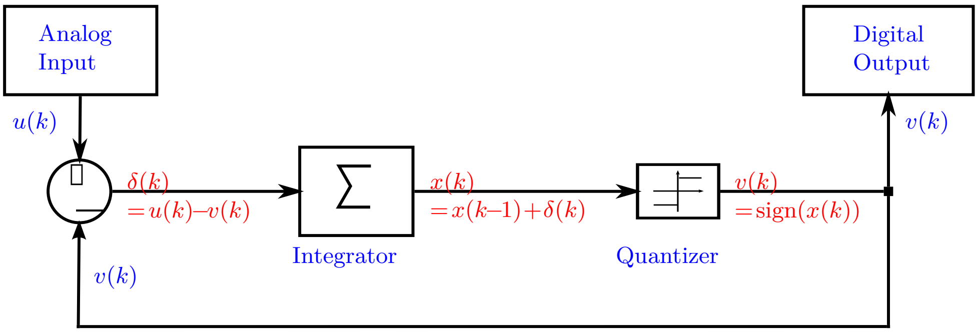
- Converts analog signal into digital signals
- Used in many electrical devices interfacing with a physical environment[.5cm]

$\Delta - \Sigma$ *modulators*

- Widely used family of ADCs
- Efficient processing of the *quantization error*, i.e., the difference between the analog input and the digital output

A Simple Discrete-Time $\Delta - \Sigma$ Modulator

Principle Control of quantization error using a feedback loop



- The *quantization error* is the difference between the input and the output
- The *integrator* stores the summation of δs in a state variable x
- The *quantizer* produces the output based on the sign of x

Higher Order $\Delta - \Sigma$ Modulators

- More complex designs use more than one integrator[.3cm]
- The *order* of a $\Delta-\Sigma$ modulator is the number of integrators used[.3cm]
- Beginning from order three, a *stability* issue appears[.3cm]
- i.e. the integrators states can reach a *saturation* threshold compromising the analog to digital conversion

Experiments with a Third Order $\Delta - \Sigma$ Modulator

Outline

Contents

Questions and Existing Results

First Question
When does saturation occur?

Second Question

Does saturation always imply a bad conversion?

Existing Results

- Hybrid system model;
- Some answer to the first question for a limited horizon;
- Nothing for the second question (*Fourier transform!*).

A third order $\Delta - \Sigma$ modulator, Simulink model

- We get a stochastic system by randomly choosing the inputs $u(k)$
- State s_k is the tuple $(u(k), x_1(k), x_2(k), x_3(k), v(k))$
- The next state s_{k+1} is determined by the random choice of $u(k+1)$ and computed by the Simulink engine
- For all k, $u(k)$ is chosen uniformly in $[-u_{\max}, u_{\max}]$[.3cm]

$\Rightarrow$ Statistical analysis for all input signals of amplitude bounded by $u_{\max}$

Saturation Analysis

Probability of saturation occurrence for different values of $u_{\max}$?

- Let $Satur$ be a boolean predicate
- For all state $s = (u, x_1, x_2, x_3, v)$, let $L(s) = \{Satur\}$ iff $|x_3| \geq 1$

We can then evaluate the formula $Pr_{\geq\theta}(\Diamond Satur)$.[.3cm] A tool :

- A routine checking $\sigma \models \Diamond Satur$
- The sequential ratio testing algorithm which decides whether $\mathcal{S} \models Pr_{\geq\theta}(\phi)$ given θ, α, β and δ
- A simple bisection procedure which tries to maximize the value of θ for which the answer is true

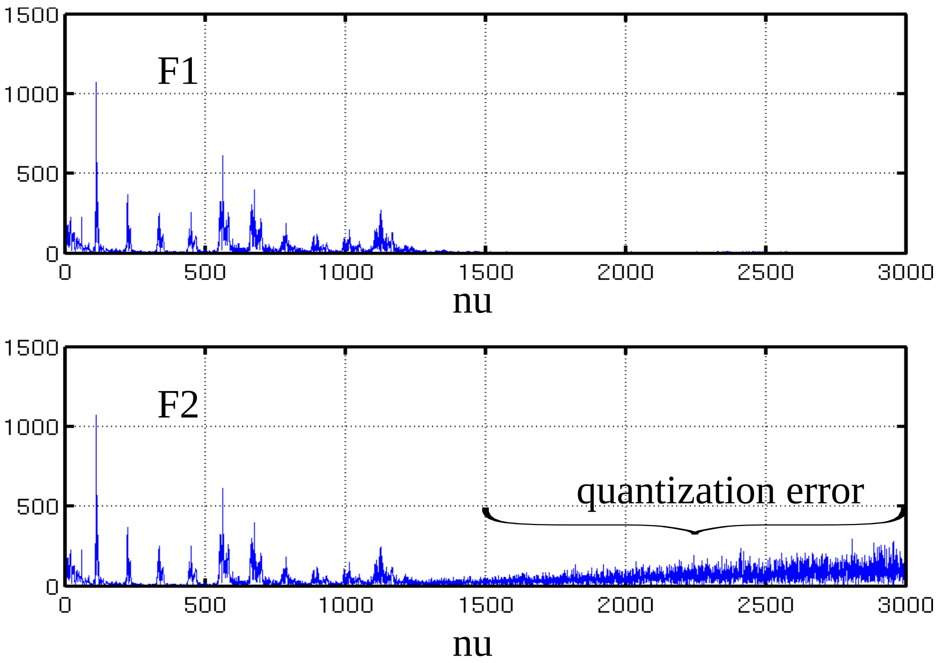
Experimental Results

$u_{\max}$	Hypothesis Accepted	Number of executions
0.1	$p \leq 0$	416
0.15	$p \geq 0.0938$	4967
0.2	$p \geq 0.640625$	17815
0.25	$p \geq 0.984375$	416
0.3	$p \geq 1$	688

Table of results for $p = Pr(\sigma \models \Diamond Satur)$, with $\alpha = \beta = 1e^{-3}$ and $\delta = 1e^{-2}$

- Consistent with results formally obtained in [Dang Donze Maler 04] but on a much larger horizon (24000 as compared to 31)
- The expected number of simulations grows logarithmically w.r.t. the inverse of α and β and polynomially w.r.t. the inverse of δ

Frequency Domain



[.5cm]

- Quantization pushes error towards high frequencies;
- Suggestion : Check for quality under small frequencies.

Execution Predicate in the Frequency Domain

- Let $F_u(\sigma)$ and $F_v(\sigma)$ be the Fourier Transforms (FTs) of the input signal associated with σ[.2cm]
- Let $d_f^{\nu_0}(\hat{\xi}_1, \hat{\xi}_2)$ be a measure of the distance between two FTs $\hat{\xi}_1$ and $\hat{\xi}_1$ for frequencies smaller than ν_0[.2cm]
- Then we can derive an execution predicate p_f such that

$$p_f(\sigma) = \mathbf{T} \text{ iff } d_f^{\nu_0}(F_u(\sigma), F_v(\sigma)) \leq \epsilon,$$

For $\nu_0 = 100Hz$ and $\epsilon \leq .1$ the predicate discriminates between "correct" and "failed" conversions

Frequency Domain Predicate, Experimental Results

$u_{\max}$	Hypothesis Accepted	Number of Executions
0.8	$p \geq 1$	688
0.9	$p \geq 0.984375$	612
1.0	$p \geq 0.984375$	1248
1.1	$p \geq 0.875$	6388
1.2	$p \geq 0.578125$	15507

Table of results for $p = Pr(p_f)$, with $\alpha = \beta = 1e^{-3}$ and $\delta = 1e^{-2}$

Experiments Interpretation

The previous results show that

- For $u_{\max} \geq 0.3$ the system satisfies $\diamondsuit Satur$ with probability 1
- For $u_{\max} \leq 0.8$ the system satisfies p_f with probability 1 [.5cm]

Thus we statistically established that for $0.3 \leq u_{\max} \leq 0.8$, the formula $\diamondsuit Satur \wedge p_f$ is satisfied with probability 1, meaning that saturation can occur without a dramatic decrease in the conversion quality[.8cm]

This extends the results in [Gupta Krogh Rutenbar 04] and [Dang Donze Maler 04] where it was conservatively assumed that the absence of saturation was necessary for a proper behavior

Conclusion and Perspectives

Conclusion on $\Delta - \Sigma$ Modulator

Summary

- A framework for the statistical probabilistic Model Checking of mixed-signal circuits
- The simulation-based approach makes it easier to deal with functionals on executions such as the Fourier transform
- Application to a non-trivial case study for which we improved previous results

Future work

- Extension to unbounded execution and dense time using appropriate monitoring techniques
- Logic mixing temporal properties and partial execution predicates
- More precise definitions and specifications for frequency domain properties based on the need of analog designers

System's Biology

Systems Biology

- In presence of a few species, reactions are defined in terms of stochastic processes;
- In such context, one wants to exercise the master equation that governs system's evolution.

  Definition 8. The master equation : phenomenological set of first-order differential equations describing the time evolution of the probability of a system to occupy each one of a discrete set of states.

Situation

- We want to Solve stochastic equations, but
- Many stochastic equations are numerically intractable!

BionetGen and Gillespie

- BionetGen Toolset:

  Very simple language to model proteins and proteins-proteins interactions :

  Uses rewriting rules like the k-calculus

- Gillespie algorithm simulates rule applications (Continuous-timed Markov Chains);

- Systems can be big : more than 6 hours for a simulation !

  $\Rightarrow$ distributed implementation.

BionetGen Language (1)

The language allows to describe

- Molecules and functional;
- States of functional;
- Binding between functional and molecules;
- Chemical reactions;
- ....

Available at

`http://bionetgen.org/index.php/Main_Page`

BionetGen Language (2)

Example 9. Molecule

```
R(l,d,Y~P)
```

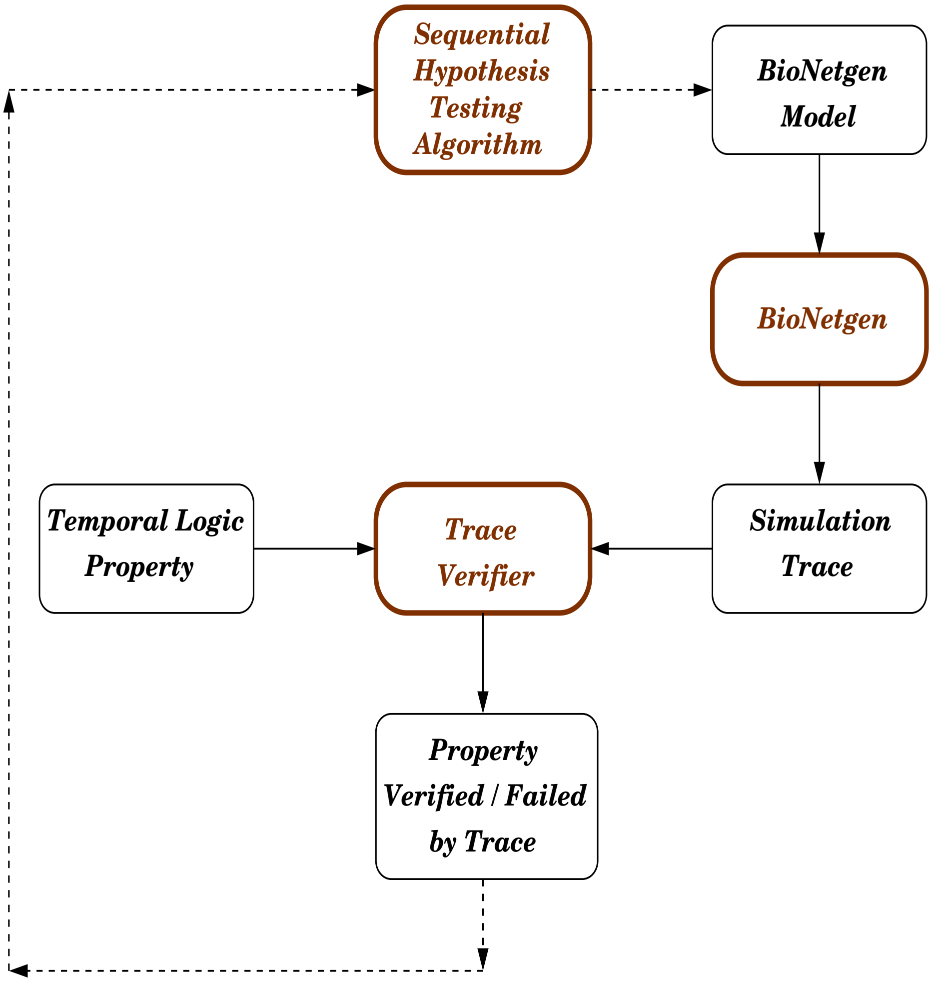
Example 10. Chemical reaction

```
L(r) + R(l,d) <-> L(r!1).R(l!1,d) kp1, km1
```

Biolab

- Combine BionetGen with SPRT;
- A logic for biologist;
- Formal validation of observations (T-Cell model, ...).

Architecture of BIOLAB



[.5cm]

The T-Cell model

- Detect antigen and should react properly;

- Should not react to non pathological proteins;
- Property : the system can alternate between reactive an nonreactive states.

6 Bayesian Model Checking

Bayesian Testing

1. Prior probability (informative Vs. non informative) on H_0 and H_1;
2. Prior information is used to decrease the number of experiments;
3. Bayesian Testing is more driven towards compound hypothesis than statistical hypothesis testing!

Future Work : Bayesian risk and nested operators.

Non Informative Prior

θ	$Bayes'\ Factor\ Test$		$SPRT$ ($\delta = 0.01$)		$SPRT$ ($\delta = 0.001$)	
	H_1	H_0	H_1	H_0	H_1	H_0
0.95	2		275		349	
0.9	8		610		608	
0.8	35*		-	-	-	-
0.7	81*		-	-	-	-
0.6	591*		-	-	-	-
0.5		272	-	-	-	-
0.4		156	-	-	-	-
0.3		16	-	-	-	-
0.2		5		909		929
0.1		9		446		468
0.05		2		201		189

7 What's next?

Next part of the tutorial :

- Model Checking PCTL* using hypothesis testing;
- Neested probability operators;
- Black-box systems.

1 Overview

Session I Overview

- Sampling + Hypothesis testing can be used to infer parameters of a Bernoulli distribution
- Application: Verification of properties $P_{\geq\theta}(\psi)$ where ψ is such that
  - for every execution σ, there is a finite prefix u such that $\sigma \models \psi$ iff $u \models \psi$, and
  - for any finite prefix u, $u \models \psi$ can be easy checked

Session II Overview

- Extend ideas of Session I to develop algorithms to verify properties in a full logic like PCTL. Main challenges include verifying properties $P_{\geq\theta}(\psi)$, where
  - Determining the satisfaction of ψ on an execution requires further statistical tests
  - Satisfaction of ψ on an execution not determined by a finite prefix, *e.g.,* $p\,U\,q$
- Use ideas of statistical model checking to verify "black-box" or "model-less" systems

Session II Outline

- Overview of Measure Spaces, Markov Chains, and PCTL
- Model checking PCTL
- Model checking black-box systems

Part I
Preliminaries

2 Measure Spaces

σ-Field

Definition 1. A *σ-field* over a set X is a collection, Σ, of subsets X such that

- $X \in \Sigma$
- If $A \in \Sigma$ then $X \setminus A \in \Sigma$, and
- If $\{A_i\}_{i\in I}$ is a countable collection of sets from Σ then $\bigcup_{i\in I} A_i \in \Sigma$

Example 2. Given a set X, the collections $\Sigma_1 = \{\emptyset, X\}$ and $\Sigma_2 = 2^X$ are examples of σ-fields.

Smallest σ-Field

- Intersection of arbitrary σ-fields is again a σ-field
- Thus, given any collection C of subsets of X, there is a unique smallest σ-field that contains C
- This is said to be the σ-field generated by C

Probability Measures

Definition 3. A *probability measure*, μ, over (X, Σ) is $\mu : \Sigma \to \mathbb{R}_{\geq 0}$ such that

- $\mu(\emptyset) = 0$
- For a countable collection of pairwise disjoint sets $\{A_i\}_{i \in I}$, $\mu(\bigcup_{i \in I} A_i) = \sum_{i \in I} \mu(A_i)$
- $\mu(X) = 1$

3 Markov Chains

Markov Chain

Definition 4. A Markov Chain over a set of propositions AP is a $\mathcal{M} = (Q, q_s, \delta, L)$, where

- Q is a set of (not necessarily finite) states,
- $q_s \in Q$ is the initial state,
- $L : Q \to 2^{AP}$ is a labelling function, and
- $\delta : Q \times Q \to [0, 1]$ is a transition function with the property that for every q, $\sum_{q' \in Q} \delta(q, q') = 1$

Measurable Sets

- A *run* ρ is an element of Q^ω. ρ starts from q if the first state in ρ is q; the collection of all such runs is denoted by $\mathrm{path}(q)$.
- For $u \in Q^*$, define $C_u = \{u \cdot \rho \mid \rho \in Q^\omega\}$
- The *measurable sets of runs* are those belonging to the smallest σ-field generated by $\{C_u \mid u \in Q^*\}$

Probability Measure Defined by Markov Chains

Definition 5. The probability measure on runs defined by $\mathcal{M} = (Q, q_s, \delta, L)$ is the unique measure, μ, satisfying the following. For $u = q_0, q_1, \ldots q_n$

$$\mu(C_u) = \prod_{i=0}^{n-1} \delta(q_i, q_{i+1})$$

Sampling executions of a Markov Chain generates runs according to this measure.

4 PCTL

PCTL Syntax

Definition 6. The formulas of PCTL over a set of atomic propositions AP are given by the following grammar

$$\varphi ::= \text{true} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid P_{\bowtie\theta}(\psi)$$

where $a \in AP$, $\bowtie \in \{<, \leq, >, \geq\}$, $\theta \in \mathbb{R}$, and ψ is a path formula given by the following grammar

$$\psi ::= X\varphi \mid \varphi U \varphi$$

PCTL Semantics State Formulas

Definition 7. Satisfaction of a state formula φ at a state q is inductively defined as follows

- $q \models \text{true}$
- $q \models a$ iff $a \in L(q)$
- $q \models \neg\varphi$ iff q does not satisfy φ
- $q \models \varphi_1 \wedge \varphi_2$ iff $q \models \varphi_1$ and $q \models \varphi_2$
- $q \models P_{\bowtie\theta}(\psi)$ iff $\mu(\{\pi \in \text{paths}(q) \mid \pi \models \psi\}) \bowtie \theta$ [1]

PCTL Semantics Path Formulas

Definition 8. Satisfaction of a path formula ψ on a path $\pi = q_0 q_1 \cdots$ is inductively defined as follows

- $\pi \models X\varphi$ iff $q_1 \models \varphi$
- $\pi \models \varphi_1 U \varphi_2$ iff there is an i such that $q_i \models \varphi_2$ and for all $j < i$, $q_j \models \varphi_1$

We say $\mathcal{M} \models \varphi$ iff $q_s \models \varphi$

PCTL Syntax Abbreviations

We will use the following abbreviations

- $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$
- $\varphi_1 U^{\leq n} \varphi_2$ says that "φ_2 holds within n steps and φ_1 holds until then"

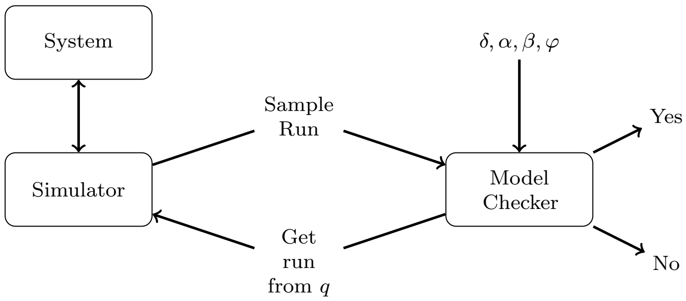
[1] For any state q and path formula ψ, $\{\pi \in \text{paths}(q) \mid \pi \models \psi\}$ is measurable.

Part II
Model Checking PCTL

5 Basic Operators

5.1 Overview

Schematic Picture



The system model could be any probabilistic model with a well understood probability space over executions that can be sampled, and a logic over that probability space.

Properties of the Algorithm

Let $\mathcal{A}^{\delta}_{\mathcal{M}}(q, \varphi, \alpha, \beta)$ be the result of the algorithm when checking if φ holds in state q in model $\mathcal{M}$, with error parameters α and β, and indifference region δ. If $\mathcal{M}$ is such that

C For every subformula of φ of the form $P_{\geq p}(\psi)$ and every state q', the measure of paths satisfying ψ is not in $[\frac{p-\delta-\alpha}{1-\alpha}, \frac{p+\delta}{1-\beta}]$

then

$$\begin{aligned} \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q, \varphi, \alpha, \beta) = \text{true} \mid q \not\models \varphi] \leq \alpha \quad \text{and} \\ \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q, \varphi, \alpha, \beta) = \text{false} \mid q \models \varphi] \leq \beta \end{aligned}$$

Algorithm Structure

```
A^δ_M(q, φ, α, β) {
    switch (φ) {
        case true:  return true
        case a ∈ AP:  return (a ∈ L(q))
        case ¬φ':  return verifyNot(q, φ, α, β)
        case φ1 ∧ φ2:  return verifyAnd(q, φ, α, β)
        case P≥p(ψ):  return verifyProb(q, φ, α, β)
    }
}
```

We cannot distinguish between strict and non-strict inequalities in $P_{\bowtie p}(\psi)$. Also, $P_{<p}(\psi)$ is logically equivalent to $\neg P_{\geq p}(\neg\psi)$.

5.2 Algorithm for Boolean Operators

Negation

```
verifyNot(q, ¬φ', α, β) {
    return ¬A^δ_M(q, φ', β, α)
}
```

Observe that (inductively)

$$\begin{aligned}
\beta &> \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi',\beta,\alpha) = \mathrm{true} \mid q \not\models \varphi'] \\
&= \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\neg\varphi',\alpha,\beta) = \mathrm{false} \mid q \models \neg\varphi'] \\
\alpha &> \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi',\beta,\alpha) = \mathrm{false} \mid q \models \varphi'] \\
&= \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\neg\varphi',\alpha,\beta) = \mathrm{true} \mid q \not\models \neg\varphi']
\end{aligned}$$

Conjunction

```
verifyAnd(q, φ1 ∧ φ2, α, β) {
    return A^δ_M(q, φ1, α, β/2) ∧ A^δ_M(q, φ2, α, β/2)
}
```

Observe that (inductively)

$$\begin{aligned}
&\mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1\wedge\varphi_2,\alpha,\beta) = \mathrm{false} \mid q \models \varphi_1\wedge\varphi_2] \\
&= \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1,\alpha,\beta/2) = \mathrm{false} \vee \mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_2,\alpha,\beta/2) = \mathrm{false} \mid q \models \varphi_1\wedge\varphi_2] \\
&\leq \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1,\alpha,\beta/2) = \mathrm{false} \mid q \models \varphi_1\wedge\varphi_2] + \\
&\quad\ \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_2,\alpha,\beta/2) = \mathrm{false} \mid q \models \varphi_1\wedge\varphi_2] \\
&= \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1,\alpha,\beta/2) = \mathrm{false} \mid q \models \varphi_1] + \\
&\quad\ \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_2,\alpha,\beta/2) = \mathrm{false} \mid q \models \varphi_2] \\
&= \beta/2 + \beta/2 = \beta
\end{aligned}$$

Conjunction (contd)

Observe that (inductively)

$$\begin{aligned}
&\mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1\wedge\varphi_2,\alpha,\beta) = \mathrm{true} \mid q \not\models \varphi_1\wedge\varphi_2] \\
&\leq \max(\mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1\wedge\varphi_2,\alpha,\beta) = \mathrm{true} \mid q \not\models \varphi_1], \\
&\qquad \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1\wedge\varphi_2,\alpha,\beta) = \mathrm{true} \mid q \not\models \varphi_2]) \\
&\leq \max(\mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_1,\alpha,\beta) = \mathrm{true} \mid q \not\models \varphi_1], \\
&\qquad \mathrm{Prob}[\mathcal{A}^{\delta}_{\mathcal{M}}(q,\varphi_2,\alpha,\beta) = \mathrm{true} \mid q \not\models \varphi_2]) \\
&= \alpha
\end{aligned}$$

6 Probabilistic Operator

6.1 Simple Formulas

Simple Formulas

Definition 9. A *simple* formula is of the form $P_{\geq p}(\psi)$, where

- ψ only uses the path operators X and $U^{\leq n}$, and
- ψ does not have any probabilistic operators

Checking Simple Formulas

To check if q satisfies a simple formula $P_{\geq p}(\psi)$, use either the single sampling plan or sequential hypothesis testing to statistically determine is the measure of paths satisfying ψ is $\geq p$ with indifference region 2δ. Draw samples as follows

> Simulate the system from q until you get a finite path that either provably satisfies ψ or provably violates ψ

6.2 Bounded Path Formulas

Bounded Path Formulas

Definition 10. A *bounded path formula* is of the form $P_{\geq p}(\psi)$, where

- ψ only uses path operators X and $U^{\leq n}$

ψ may have nested probabilistic operators.

Checking Bounded Path Formulas

Challenge
Consider a formula $P_{\geq p}(XP_{\geq p'}(Xa))$, and drawing a sample run $\rho = q, q_1, \ldots$

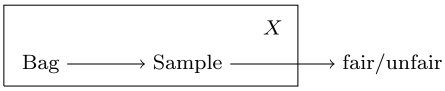
- We cannot determine if $q_1 \models P_{\geq p'}(Xa)$, and so we don't know if $\rho \models XP_{\geq p'}(Xa)$
- We can statistically determine if $q_1 \models P_{\geq p'}(Xa)$. How do we account for the error in the estimation?

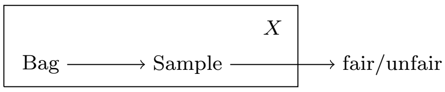
Proportion of Fair Coins

Problem
Given a bag of coins, are most (75%) of the coins fair? If the coins can be physically examined to fair or not, then we have the following situation We can test the random variable X statistically to determine if $\text{Prob}[X = \text{fair}] \geq 0.75$.

Proportion of Fair Coins

If we cannot physically examine the coins to determine if they will be fair, we still want to test random variable X below. But we observe the following random experiment Y





Relating X and Y

Let X be Bernoulli with parameter p_x and Y be Bernoulli with parameter p_y. Suppose the test has errors (α, β), we have

$$\begin{array}{l} \mathrm{Prob}[Y = \mathrm{fair} \mid X \neq \mathrm{fair}] \leq \alpha \\ \mathrm{Prob}[Y \neq \mathrm{fair} \mid X = \mathrm{fair}] \leq \beta \end{array}$$

Relating X and Y

Let X be Bernoulli with parameter p_x, Y be Bernoulli with parameter p_y, and test have error (α, β). Then,

$$\begin{aligned} p_y &= \mathrm{Prob}[Y = \mathrm{fair}] \\ &= \mathrm{Prob}[Y = \mathrm{fair} \mid X \neq \mathrm{fair}]\mathrm{Prob}[X \neq \mathrm{fair}] \\ &\qquad + \mathrm{Prob}[Y = \mathrm{fair} \mid X = \mathrm{fair}]\mathrm{Prob}[X = \mathrm{fair}] \\ p_y &\leq \alpha(1 - p_x) + 1.p_x \\ p_y &\geq (1 - \beta)p_x \end{aligned}$$

as $\mathrm{Prob}[Y = \mathrm{fair} \mid X = \mathrm{fair}] \geq 1 - \beta$. This means if $p_x < \frac{p-\delta-\alpha}{1-\alpha}$ then $p_y < p - \delta$ and if $p_x > \frac{p+\delta}{1-\beta}$ then $p_y > p + \delta$. Thus, we sample from Y and test the sample against p with indifference region 2δ.

Algorithm for Bounded Path Formulas

```
verifyProb(q, P_{≥p}(ψ), α, β) {
    Do as in simple hypothesis testing of Bernoulli variable
    except when drawing a sample do the following ...
        Get sample run π from q
        switch (ψ) {
            case Xφ':  return A^δ_M(π[1], φ', β, α)
            case φ1 U^{≤n} φ2:  return verifyBUntil(q, ψ, α, β)
        }
}
```

Bounded Until

To check if π satisfies $\varphi_1 U^{\leq n} \varphi_2$, we check if there is an $i \leq n$, such that $\pi[i] \models \varphi_2$, and $\pi[j] \models \varphi_1$, for $j < i$. We do these tests statistically, giving us the following test.

```
verifyBUntil(q, φ1 U^{≤n} φ2, α, β) {
    for i = 0 to n do
        if A^δ_M(π[i], φ2, β/n, α) then return true
        else if not A^δ_M(π[i], φ1, β/n, α) then return false
}
```

$$\boxed{\begin{array}{r} Y \\ \text{Bag} \longrightarrow \text{Sample} \longrightarrow \text{Test} \longrightarrow \end{array}} \text{fair/unfair}$$

Justification for parameters is same as for conjunction.

6.3 Unbounded Until

Unbounded Until

When drawing samples for bounded path formulas we know when to stop simulating

- when checking $X\varphi'$, we simulate the system for one step
- when checking $\varphi_1 \, U^{\leq n} \, \varphi_2$, we simulate for at most n steps

But for $\psi = \varphi_1 \, U \, \varphi_2$ we don't know when to stop simulation; there maybe no finite prefix that determines the unsatisfiability ψ on a run.

Stopping Probability

We will modify the simulation as follows: at every step, either stop simulation with probability p_s, or continue the simulation with probability $1 - p_s$. Formally, given $\mathcal{M} = (Q, q_s, \delta, L)$, take $\mathcal{M}' = (Q \cup \{q_\perp\}, q_s, \delta', L')$ where

- $L'(q) = L(q)$ for $q \in Q$, and $L'(q_\perp)$ is such that $q_\perp \not\models \varphi_2$
- For every $q, q' \in Q$, $\delta'(q, q_\perp) = p_s$ and $\delta'(q, q') = (1 - p_s)\delta(q, q')$, and $\delta'(q_\perp, q_\perp) = 1$

Relating the two Markov Chains

Proposition 11. *Let the measure of paths in $\mathcal{M}$ from $q \in Q$ satisfying $\varphi_1 \, U \, \varphi_2$ be denoted by $p^{\mathcal{M}}$ and the measure in $\mathcal{M}'$ be denoted by $p^{\mathcal{M}'}$. If N is the number of states in $\mathcal{M}$ then*

$$p^{\mathcal{M}}(1 - p_s)^N \leq p^{\mathcal{M}'} \leq p^{\mathcal{M}}$$

Proof.

- The measure of paths from q satisfying $\varphi_1 \, U \, \varphi_2$ is obtained by solving a system of equations, through (say) Gaussian elimination
- Suppose the measure of paths from q is the ith variable solved
- By induction on i, that $p^{\mathcal{M}}(1 - p_s)^i \leq p^{\mathcal{M}'} \leq p^{\mathcal{M}}$ □

Checking Unbounded Until Formulas

- Sample finite paths from $\mathcal{M}'$
- Using the relationship between the Markov Chains $\mathcal{M}$ and $\mathcal{M}'$ setup the hypothesis testing with appropriate indifference regions like the case of nested probabilistic operators

Discussion of Unbounded Until Checking

- The algorithm presented depends on the number of states N, and the stopping probability need to make the conditions workout can be small
- [SVA 05] Another algorithm with possibly better sample performance is as follows
  - Use a special algorithm to check if a state q satisfies $P_{=0}(\varphi_1\, U\, \varphi_2)$, by drawing samples from $\mathcal{M}'$
  - Draw samples from $\mathcal{M}$ by stopping either when state satisfying φ_2 is encountered or when a state satisfying $P_{=0}(\varphi_1\, U\, \varphi_2)$ is encountered

An Alternate Statistical Approach

- The correctness guarantees of the statistical model checker only apply when the measure of paths satisfying the path subformulas are bounded away from the threshold to which they are compared.
- The basic test for a probabilistic operator tests the hypothesis $H_0:\ p' \geq p+\delta$ against the hypothesis $H_1:\ p' \leq p-\delta$
- An alternate approach [SVA04,You06] does two comparisons: (a) the hypothesis $H_0^1:\ p' \geq p+\delta$ against $H_1^1:\ p' \leq p$, and (b) the hypothesis $H_0^2:\ p' \leq p-\delta$ against $H_1^2:\ p' \geq p$
  - If H_0^1 is accepted over H_1^1 then we say $P_{\geq p}(\psi)$ holds
  - If H_0^2 is accepted over H_1^2 then we say $P_{\geq p}(\psi)$ does not hold
  - If neither of the above cases happen, the algorithm says "unknown"
- The basic probability test can be extended to all of PCTL

Beyond PCTL and Markov Chains

- The statistical model checking approach can be applied to any situation where the model's probability space, simulation algorithm, and specification logic are intrinsically tied, not just Markov Chains and PCTL
- Similar ideas have been used to analyze "real-time" models like CTMC, SMC against CSL specifications
- The approach has also been used to check properties based on FFTs (Session I)

Part III
Model Checking Black-Box Systems

7 Introduction

7.1 Motivation

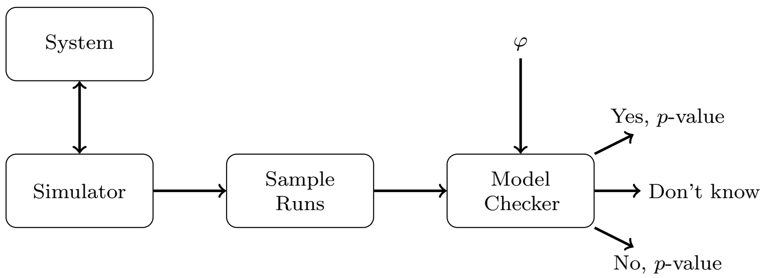
Black-Box Systems

Generating samples from any state of the system as desired maybe unreasonable in certain situations.

- When monitoring/observing a remote system over the network
- When analyzing third-party code

7.2 Problem Setup

Black-Box Systems: Schematic Picture



Limitations Imposed by Problem Setting

- Since the sample runs are drawn independent of the verification process, the algorithm cannot guarantee the correctness of its result to be within given error bounds.
- Instead, the algorithm will compute a qualitative measure of the confidence in its answer (*p-value*)
- The sample may not contain "statistical witness" for the satisfaction or violation of a property; the algorithm answers "don't know" in such cases

Comments about the Algorithm

- Runs will be assumed to be generated from a "Markovian" process; so suffix of run starting from a state q are faithful samples drawn from path(q)
- Useful results can only be obtained if the sample contains sufficiently many runs from each "relevant state"; thus, the model have finitely many "states" like a DTMC, CTMC
- Since the number of sample runs is finite, and each run is finite, unbounded until operators are essentially bounded until operators

8 The Algorithm

8.1 Probabilistic Operators

Algorithm Structure

```
verifyAtState(q, φ) {
    switch (φ) {
        case true:  return (true, 0)
        case a ∈ AP:  return ((a ∈ L(q)), 0)
        case ¬φ':  return verifyNot(q, φ)
        case φ1 ∧ φ2:  return verifyAnd(q, φ)
        case P≥p(ψ):  return verifyProb(q, φ)
    }
}
```

Algorithm returns a result (true, false, or unknown) and a p-value

Non-nested Probabilistic Operator: Observations

Suppose we want to check if q satisfies $P_{\geq p}(\psi)$. The n sample runs from q fall into 3 categories

- Those that satisfy ψ; let there be $n_\top$ such runs
- Those that satisfy $\neg\psi$; let there be $n_\bot$ such runs
- Those that satisfy neither ψ nor $\neg\psi$. This happens when short. Let there be $n_?$ such runs

Thus, the sum of all positive observations is at least $n_\top$, and at most $n - n_\bot$.

Non-nested Probabilistic Operators: Algorithm

Let X be the random variable denoting drawing a path from q that satisfies ψ, and let p' be its parameter

- If $n_\top > np$ then we say q satisfies $P_{\geq p}(\psi)$ and the confidence in the answer is bounded by $\mathrm{Prob}[\sum X \geq n_\top \mid p' = p]$
- If $n - n_\bot < np$ then we q does not satisfy $P_{\geq p}(\psi)$ and the confidence is bounded by $\mathrm{Prob}[\sum X \leq n - n_\bot \mid p' = p]$
- Otherwise, we say "don't know"

Nested Probabilistic Operators

- Once again the situation can be modelled as one where instead of observing a random variable X with parameter p_x, the sample provides evidence for another random variable Y with parameter p_y
- If α is the p-value associated with Y, we can bound p_y as $p_x - \alpha p_x \leq p_y \leq p_x + (1 - p_x)\alpha$
- Using these bounds, we can bound the confidence as $\mathrm{Prob}[\sum Y > n_\top \mid p_y = p - \alpha p]$ or $\mathrm{Prob}[\sum Y < n - n_\bot \mid p_y = p + (1 - p)\alpha]$

Nested Probabilistic Operators: Algorithm

```
verifyProb(q, P_{≥p}(ψ)) {
    max = 0; min = 0; α = 0;
    for each sample path π starting at q {
        (y, α') = verifyPath(π, ψ);
        if y = don't know then
            max = max+1
        else min = min+y; max = max+y;
        α = max(α, α')
    }
    if (min> p + (1 − α)p) then
        return (true, Prob[∑Y ≥ min | p_y = p + (1 − α)p])
    else if (max< p − αp) then
        return (true, Prob[∑Y ≤ max | p_y = p − αp])
    else return (don't know, 0)
}
```

8.2 Boolean Operators

Negation

```
verifyNot(q, ¬φ') {
    (y, α) = verifyState(q, φ')
    return (¬y, α)
}
```

Conjunction

- If q satisfies φ_1 with p-value α_1 and satisfies φ_2 with p-value α_2 then q satisfies $\varphi_1 \wedge \varphi_2$ with p-value $\max(\alpha_1, \alpha_2)$
- If q does not satisfy φ_1 with confidence α_2 (or φ_2 with α_2) then q does not satisfy $\varphi_1 \wedge \varphi_2$ with confidence α_1 (α_2)
- If q does not satisfy φ_1 and φ_2 with confidence α_1 and α_2, respectively, then q does not satisfy $\varphi_1 \wedge \varphi_2$ with confidence $\min(\alpha_1, \alpha_2)$

Extensions

- Ideas used to check PCTL properties can easily be extended to check properties in CSL

Conclusions

- Statistical hyposthesis testing can be used to verify systems in a model independent way, against a variety of properties
- The techniques have been used in a few case studies (including those discussed in Session I)
- There have also been some examples analyzed to get a better sense of the samples needed, and how the approach compares to more traditional numerical based approach