SReach: A Bounded Model Checker for Stochastic Hybrid Systems

Qinsi Wang
Computer Science
Department
Carnegie Mellon University
qinsiw@cs.cmu.edu

Paolo Zuliani
School of Computing Science
Newcastle University
paolo.zuliani@ncl.ac.uk

Soonho Kong
Computer Science
Department
Carnegie Mellon University
soonhok@cs.cmu.edu

Sicun Gao
CSAIL
Massachusetts Institute of Technology
sicung@csail.mit.edu

Edmund M. Clarke
Computer Science
Department
Carnegie Mellon University
emc@cs.cmu.edu

ABSTRACT
In this paper we describe a new tool, SReach, which solves probabilistic bounded reachability problems for two classes of stochastic hybrid systems. The first one is (nonlinear) hybrid automata with parametric uncertainty. The second one is probabilistic hybrid automata with additional randomness for both transition probabilities and variable resets. Standard approaches to reachability problems for linear hybrid systems require numerical solutions for large optimization problems, and become infeasible for systems involving both nonlinear dynamics over the reals and stochasticity. Our approach encodes stochastic information by using random variables, and combines the randomized sampling, a \( \delta \)-complete decision procedure, and statistical tests. SReach utilizes the \( \delta \)-complete decision procedure to solve reachability problems in a sound manner, i.e., it always decides correctly if, for a given assignment to all random variables, the system actually reaches the unsafe region. The statistical tests adapted guarantee arbitrary small error bounds between probabilities estimated by SReach and real ones. Compared to standard simulation-based methods, our approach supports non-deterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem. We demonstrate our method’s feasibility by applying SReach to three representative biological models and to additional benchmarks for nonlinear hybrid systems with multiple probabilistic system parameters.

1. INTRODUCTION
Stochastic hybrid systems (SHSs) are dynamical systems exhibiting discrete, continuous, and stochastic dynamics. Due to their generality, SHSs have been widely used in various areas, including cyber-physical systems, financial decision problems, and biological systems \[7,11\]. The popularity of SHSs in real-world applications motivates researchers to put a significant effort into analysis methods for this class of systems. One of the elementary questions for the quantitative analysis of SHSs is the probabilistic reachability problem, i.e., computing the probability of reaching a certain set of states. The set may represent unsafe states which should be avoided or visited only with some small probability, or dually, good states which should be visited frequently. There are two reasons why this kind of problem catches the researchers’ attention. One is that most temporal properties can be reduced to reachability problems, considering the very expressive hybrid modeling framework. The other is that probabilistic state reachability is a hard and challenging problem which is undecidable in general.

To describe stochastic dynamics, uncertainties have been added to hybrid systems in a number of different ways. One of the simplest ways replaces some of the system parameters with random variables, resulting in general hybrid automata (GHA) with parametric uncertainty. Another approach integrates deterministic flows with probabilistic jumps. When state changes forced by continuous dynamics involve discrete random events, we refer to such systems as probabilistic hybrid automata (PHAs) \[30\]. When state changes also involve continuous probabilistic events, we call this kind of models stochastic hybrid automata (SHAs) \[15\]. Other models describe randomness by substituting deterministic flows with stochastic ones, such as stochastic differential equations (SDEs) \[5\], where the random perturbation affects the dynamics continuously. When all such modifications have been applied, the resulting models are called general stochastic hybrid systems (GSHS) \[23\]. Among these different models, of particular interest for this paper are GHA with parametric uncertainty and PHA with additional randomness for both transition probabilities and variable resets.

When modeling real-world systems using hybrid models, parametric uncertainty arises naturally. Although its cause is multifaceted, two factors are critical. First, probabilistic parameters are needed when the physics controlling the system is known, but some parameters are either not known precisely, are expected to vary because of individual differences, or may change by the end of the system’s operational
lifetime. Second, system uncertainty may occur when the model is constructed directly from experimental data. Due to imprecise experimental measurements, the values of system parameters may have ranges of variation with some associated likelihood of occurrence. Clearly, the GHAs with parametric uncertainty are suitable models considering these major causes. Note that, in both cases, we assume that the probability distributions of probabilistic system parameters are known. Another interesting and more expressive class of models is PHAs, which extends hybrid automata \[21\] with discrete probability distributions. More precisely, for discrete transitions in a model, instead of making a purely nondeterministic choice over the set of currently enabled jumps, a PHA nondeterministically chooses among the set of recently enabled discrete probability distributions, each of which is defined over a set of transitions. Although randomness is defined to only influence the discrete dynamics of the model, PHAs are still very useful and have interesting practical applications \[31\]. In this paper, we consider a variation of PHAs, where additional randomness for both transition probabilities and resets of some system variables are allowed. In other words, in terms of the randomness for jump probabilities, we mean that the probabilities attached to probabilistic jumps from one mode, instead of having a discrete distribution with predefined constant probabilities, can be expressed by equations involving random variables whose distributions can be either discrete or continuous. This extension is motivated by the fact that some transition probabilities can vary due to factors such as individual and environmental differences in real-world systems. When it comes to the randomness of variable resets, we allow that a system variable can be reset to a value obtained according to a known discrete or continuous distribution, instead of being assigned with a fixed value. For example, with this extension, on a discrete update, variable \( t \) can be assigned to any value between 1 and 2 with equal probability.

In this paper, we describe our tool SReach which supports probabilistic bounded reachability analysis for these two interesting model classes: GHAs with parametric uncertainty and PHAs with additional randomness. It combines the recently proposed \( \delta \)-complete bounded reachability analysis technique \[17\] with statistical testing techniques. Our technique saves the virtues of the Satisfiability Modulo Theories (SMT) based Bounded Model Checking (BMC) for GHAs \[12,33\], namely the fully symbolic treatment of hybrid state spaces, while advancing the reasoning power to probabilistic models and requirements. By utilizing the \( \delta \)-complete analysis method, the full nondeterminism of models can be considered. By adapting statistical tests, SReach can place arbitrarily small error bounds on the estimated probabilities. Compared to standard simulation-based approaches, our approach supports nondeterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem which is critical for simulation-based methods. Comparing to the existing tools introduced in \[13,16,25,36\], besides offering a sound way to analyze nonlinear dynamics within the SHSs, SReach also supports probabilistic bounded reachability analysis for hybrid systems with parametric uncertainty. With this modeling formalism, important elements such as probabilistic initial conditions and random variable coefficients can all be expressed by multiple random variables. Furthermore, for PHAs, SReach considers a more general and useful formalism where general randomness for transition probabilities and variable resets are allowed. We discuss three biological models - a cardiac atrial fibrillation model, a prostate cancer treatment model, and our synthesized Killered biological model - to show how SReach can be used to answer several types of questions including model validation, parameter estimation, and sensitivity analysis. To further demonstrate the feasibility of SReach, we also apply it to additional real-world hybrid systems with parametric uncertainty, e.g. the quadcopter stabilization control.

**Related Work.** Analysis approaches for GSFSs are often based on Monte-Carlo simulation \[8\]. Considering the difficulty in dealing with this general case, efforts have been mainly placed on different subclasses. For PHAs, Zhang et al. \[36\] abstracted the original PHA to a probabilistic automaton (PA), and then used established Model Checking methods (e.g. PRISM \[25\]) for the abstracted model. Hahn et al. also discussed an abstraction-based method where the given PHA was translated into a \( n \)-player stochastic game using two different abstraction techniques \[21\]. Another method proposed is an SMT-based BMC procedure \[16\]. In \[1,4\], a similar class of models called discrete-time stochastic hybrid systems (DTSHSs), which is widely used in control theory, was considered. With regard to system analysis, the control problem is to find an optimal control strategy that minimizes the probability of reaching unsafe states. Zuliani et al. also mentioned a simulation-based method for model checking DTSHSs against bounded temporal properties \[37\]. We refer to this method as Statistical Model Checking (StatMC). Although StatMC does not belong to the class of exhaustive state-space exploration methods, it usually returns results faster than the exhaustive search with a predefined arbitrarily small error bound on the estimated probability. StatMC was recently integrated into UPPAAL \[27\] in order to handle very general networks of SHAs \[13\]. To analyze reachability problems of SHAs, Fränzle et al. \[15\] first over-approximated a given SHA by a PHA, and then exploited the verification procedure introduced in \[36\] to model check the over-approximating PHA. Platzer introduced another interesting modeling formalism - stochastic hybrid programs (SHPs) in \[29\]. To specify system properties, Platzer proposed a logic called stochastic differential dynamic logic, and then suggested a proof calculus to verify logical properties of SHPs.

The paper proceeds by first, in Section 2, introducing two modeling formalisms of SHSs under consideration: GHAs with parametric uncertainty, and PHAs with additional randomness. Section 3 explains how SReach solves the probabilistic bounded reachability problem by encoding stochastic dynamics and combining SMT-based BMC with statistical tests. Case studies and additional experiments are discussed in Section 4. Section 5 concludes the paper.

2. STOCHASTIC HYBRID MODELS

Before discussing the details of the SReach algorithm, we first define the two types of formalism that SReach considers. The first class is GHAs with parametric uncertainty, and PHAs with additional randomness. Section 3 explains how SReach solves the probabilistic bounded reachability problem by encoding stochastic dynamics and combining SMT-based BMC with statistical tests. Case studies and additional experiments are discussed in Section 4. Section 5 concludes the paper.

**Definition (Hybrid Automata with Parametric Uncertainty)** A hybrid automaton with probabilistic parameters is a tuple \( H_p = ((Q, E), V, RV, \text{Init, Flow, Inv, Jump, } \Sigma) \), where

- \((Q, E)\) is a finite directed multigraph. The vertices
$Q = \{q_1, \cdots, q_m\}$ is a finite set of discrete modes, and edges in $E$ are control switches.

- $V = \{v_1, \cdots, v_n\}$ denotes a finite set of real-valued system variables, where $n$ is the dimension of $H$. We write $\bar{V}$ for the set $\{\bar{v}_1, \cdots, \bar{v}_n\}$ to represent first derivatives of variables during the continuous change, and write $V'$ for the set $\{v_1', \cdots, v_n'\}$ to denote values of variables at the conclusion of the discrete change.

- $RV = \{v_1, \cdots, v_k\}$ is a finite set of random variables, where the distribution of $u_i$ is denoted by $P_i$.

- $\text{Init}$, $\text{Flow}$, and $\text{Inv}$ are labeling functions over each mode $q \in Q$. The initial condition $\text{Init}(q)$ is a predicate whose free variables are from $V \cup RV$, the invariant condition $\text{Inv}(q)$ is a predicate whose free variables are from $V \cup RV$, and the flow condition $\text{Flow}(q)$ is a predicate whose free variables are from $V' \cup RV$.

- $\Sigma$ is a finite set of events, and an edge labeling function $\text{event} : E \rightarrow \Sigma$ assigns to each control switch an event.

$S\text{Reach}$ also considers PHAs with additional randomness formally defined as follows.

**Definition (Probabilistic Hybrid Automata)** A probabilistic hybrid automaton (with additional randomness) $H$ is a tuple $(Q, \bar{q}, V, \{\text{Post}_i\}_{i \in M}, RV, Cnds)$ where

- $Q := \{q_1, \cdots, q_n\}$ is a finite set of control modes.
- $\bar{q} \subseteq Q$ is the initial mode.
- $V = \{v_1, \cdots, v_k\}$ denotes a finite set of real-numbered system variables, where $k$ is the dimension of $H$. As mentioned, $V$ represents first derivatives of variables, and $V'$ denotes values of variables at the conclusion of the discrete change.
- $\{\text{Post}_i\}_{i \in Q}$ indicates continuous-time behaviors on each mode.
- $RV$ is a finite set of random variables with known discrete or continuous probability distributions.
- $Cnds$ is a finite set of probabilistic guarded commands of the form: $g \rightarrow p_1 : v_1 + \cdots + p_m : u_m$, where $g$ is a predicate representing a transition guard with free variables from $V$, $p_i$ is the transition probability for the $i$th probabilistic choice which can be expressed by an equation involving random variable(s) in $RV$ and the $p_i$’s satisfy $\sum_{i=1}^{m} p_i = 1$, and $u_i$ is the corresponding transition updating function for the $i$th probabilistic choice, whose free variables are from $V \cup V' \cup RV$.

To illustrate the additional randomness allowed for transition probabilities and variable resets, an example probabilistic guarded command is $x \geq 5 \rightarrow p_1 : (x' = \sin(x)) + (1 - p_1) : (x' = p_2)$, where $x$ is a system variable, $p_1$ has a Uniform distribution $U(0.2, 0.9)$, and $p_2$ has a Bernoulli distribution $B(0.85)$. This means that, the probability to choose the first transition is not a fixed value, but a random one having a Uniform distribution. Also, after taking the second transition, $x$ can be assigned to either 1 with probability 0.85, or 0 with 0.15. In general, for an individual probabilistic guarded command, the transition probabilities can be expressed by equations of one or more new random variables, as long as values of all transition probabilities are within $[0, 1]$, and their sum is 1. Currently, all four primary arithmetic operations are supported. Note that, to preserve the Markov property, only unused random variables can be adapted, so that no dependence between the current probabilistic jump and previous transitions will be introduced.

### 3. SReach Algorithm

First, $S\text{Reach}$ uses a set of random variables to encode all the stochastic information. In detail, when a hybrid automaton with parametric uncertainty is given, $S\text{Reach}$ directly declares each probabilistic system parameter as a random variable with a known distribution. While for a PHA, each probabilistic guarded command $g \rightarrow p_1 : v_1 + \cdots + p_m : u_m$ is rewritten by introducing a new random variable $rv$ such that $Pr(rv = i) = p_i$. For example, a probabilistic command $x \geq 1 \rightarrow 0.7 : (x' = 1) + 0.3 : (x' = x)$ will be rewritten as two new guarded commands after introducing a new random variable $r$ whose distribution is $(Pr(r = 1) = 0.7, Pr(r = 2) = 0.3)$. The first command is $x \geq 1 \land r = 1 \rightarrow x' = 1$. The second is $x \geq 1 \land r = 2 \rightarrow x' = x$. When additional randomness is involved in assigning probabilities for probabilistic transitions or in resetting system variables, extra random variables are needed. For instance, $S\text{Reach}$ can express a probabilistic guarded command as $x \geq 1 \rightarrow p_1 : (x' = p_2) + (1 - p_1) : (x' = x)$, where $p_1$ is a random variable which obeys a Uniform distribution $U(0.6, 0.85)$, and $p_2$ is a random variable whose distribution is $N(0, 1)$, i.e., normal with mean 0 and standard deviation 1.

After encoding all the stochastic elements using random variables, $S\text{Reach}$ randomly samples all the random variables according to their probability distributions. For each sampled assignment to these random variables, we obtain a corresponding intermediate hybrid automaton by replacing all the random variables with their assigned values. Then, the $\delta$-complete analyzer $d\text{Reach}$ [17] is utilized to analyze each intermediate hybrid automaton $M_i$, together with the desired precision $\delta$ and unfolding depth $k$. The analyzer returns either unsat or $\delta$-sat for $M_i$ (see Appendix [8] for more details on $\delta$-complete decision procedures). This information is then used by statistical tests to decide whether to stop or to repeat the procedure, and to return the estimated probability. The full procedure is illustrated in Algorithm [4] where $MP$ is a given probabilistic model, and $ST$ indicates which statistical testing method will be used. $Succ$ is used to record the number of $\delta$-sat instances that are returned by $d\text{Reach}$, and $N$ denotes the total number of samples generated so far. These two numbers are then the inputs of $S\text{Reach}$’s statistical testing procedure $ST$. The procedure $ExtractRV$ is used to obtain the full set of involved random variables, $Sim$ is to sample them according to their probability distributions, and $Gen$ is to generate an intermediate hybrid automaton considering the original probabilistic model and a sampled assignment to random variables. Since the full nondeterminism within the intermediate hybrid automata has been considered when handling the bounded reachability problems, the estimated probabilities computed by $S\text{Reach}$ are the maximum probabilities. Also, for a probabilistic hybrid automaton, sampling and fixing all the probabilistic transitions in advance results in an over-approximation of the original probabilistic model. Because the result is an over-approximation, safety properties are preserved. To improve the performance of $S\text{Reach}$, each sampled assignment, together with its corresponding $d\text{Reach}$ result, has been recorded for avoiding repeated calls.
to dReach with the same sampled assignments. This significantly reduces the total calls to dReach for PHAs (with additional randomness for transition probabilities), as the size of the sample space involving random variables describing probabilistic jumps is comparatively small. For example the PHA, as shown in Figure 1 with this improvement, the total checking time for a reachability problem has been decreased from 11291.31s for 658 samples (17.16s per sample) to 3295.82s (5.01s per sample). To further improve the performance, a parallel version of SReach has been implemented using OpenMP, where multiple samples and corresponding hybrid automata are generated, and passed to dReach simultaneously. Using this parallel SReach on a 4-core machine, the running time for the example PHA has been further decreased to 2119.55s for 660 samples (3.33s per sample).

Algorithm 1 SReach

1: function SREACH(MP, ST, δ, k)
2: Succ ← 0
3: N ← 0
4: RV ← ExtractRV(MP)
5: repeat
6:   Si ← Sim(RV)
7:   Mi ← Gen(MP, Si)
8:   Res ← dReach(Mi, δ, k)
9:   if Res = δ-sat then
10:      Succ ← Succ + 1
11: end if
12: N ← N + 1
13: until ST.done(Succ, N)
14: return ST.output
15: end function

Currently, SReach supports a number of hypothesis testing and statistical estimation techniques including: Lai’s test [26], Bayes factor test [24], Bayes factor test with indifference region [35], Sequential probability ratio test (SPRT) [54], Chernoff-Hoeffding bound [22], Bayesian Interval Estimation with Beta prior [57], and Direct Sampling. All methods produce answers which are correct up to a precision that can be set arbitrarily by the user. See Appendix A for more details about how these tests can guarantee an arbitrary small error bound between the estimated probability and the real one. With these hypothesis testing methods, SReach can answer qualitative questions, such as “Does the model satisfy a given reachability property in k steps with probability greater than a certain threshold?” With the above statistical estimation techniques, SReach can offer answers to quantitative problems. For instance, “What is the probability that the model satisfies a given reachability property in k steps?” SReach can also handle additional types of interesting problems by encoding them as bounded reachability problems.

The model validation problem with prior knowledge can be encoded as a bounded reachability question. After expressing prior knowledge about the given model as reachability properties, is there any number of steps k in which the model satisfies a given property? If none exists, the model is incorrect regarding the given prior knowledge. If, for each property, a witness is returned, we can conclude that the model is correct with regard to the prior knowledge. The parameter estimation problem can also be encoded as a k-step reachability problem. Does there exist a parameter combination for which the model reaches the given goal region in k steps? If so, this parameter combination is potentially a good estimation for the system parameters. The goal here is to find a combination with which all the given goal regions can be reached in a bounded number of steps. Moreover, sensitivity analysis can be conducted by a set of bounded reachability queries as well: Are the results of reachability analysis the same for different possible values of a certain system parameter? If so, the model is insensitive to this parameter with regard to the given prior knowledge.

4. EXPERIMENTS

Our method is implemented in the open-source tool SReach (https://github.com/dreal/SReach). Both a sequential version and a parallel one have been implemented. See Appendix C for information on using SReach. All models for the following case studies and additional benchmarks can be found on the tool website. All experiments were conducted on a server with 2* AMD Opteron(tm) Processor 6172 (24 cores) and 32GB RAM, running on Ubuntu 14.04.1 LTS. 12 cores were used. In our experiments we used 0.001 as the precision for the δ-decision problem, and Bayesian sequential estimation with 0.01 as the estimation error bound, coverage probability 0.99, and a uniform prior (α = β = 1).

Atrial Fibrillation. The minimum resistor model (MRM) reproduces experimentally measured characteristics of human ventricular cell dynamics [10]. The MRM reduces the complexity of existing models by representing channel gates of different ions with one fast channel, and two slow gates. However, due to this reduction, for most model parameters, it becomes impossible to obtain their values through measurements. After adding parametric uncertainty into the original hybrid model, we show that SReach can be adapted to estimate parameters for this stochastic model, i.e., identifying appropriate ranges and distributions for model parameters. To illustrate the way in which SReach is used to conduct parameter estimation, we chose two system parameters - EPI1TO1 and EPI1TO2, and varied their distributions to see which ones allow the model to present the desired pattern. Each model has 4 modes. In the experiments, we chose 3 as the unfolding depth. For each sample generated, SReach analyzed systems with 62 variables, and 24 ODEs. As in Table 1 when EPI1TO1 is either close to 400, or between 0.0061 and 0.007, and EPI1TO2 is close to 6, the model can satisfy the given bounded reachability property with a probability very close to 1.

Prostate cancer treatment. This model is a nonlinear hybrid automaton with parametric uncertainty. We modified the model of the intermittent androgen suppression (IAS) therapy in [32] by adding parametric uncertainty. The IAS therapy switches between treatment-on, and treatment-off with respect to the serum level thresholds of prostate-
Table 1: Results for the atrial fibrillation model. #RVs = number of random variables in the model, #S = number of δ-sat samples, #T = total number of samples, Est_P = estimated maximum posterior probability, A_T(s) = average CPU time of each sample in seconds, and Tot_T(s) = total CPU time for all samples in seconds.

<table>
<thead>
<tr>
<th>#S</th>
<th>#T</th>
<th>Est_P</th>
<th>A_T(s)</th>
<th>Tot_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>6</td>
<td>0.996</td>
<td>0.270</td>
<td>64.60</td>
</tr>
<tr>
<td>0</td>
<td>240</td>
<td>0.004</td>
<td>0.042</td>
<td>10.08</td>
</tr>
<tr>
<td>5(0.1, 0.129)</td>
<td>0</td>
<td>240</td>
<td>0.004</td>
<td>0.038</td>
</tr>
<tr>
<td>N(300, 1e-4)</td>
<td>N(6, 1e-4)</td>
<td>240</td>
<td>0.996</td>
<td>0.091</td>
</tr>
<tr>
<td>N(5.5e-3, 10e-6)</td>
<td>N(0.11, 10e-5)</td>
<td>0</td>
<td>240</td>
<td>0.004</td>
</tr>
</tbody>
</table>

Specific antigen (PSA), namely r_0 and r_1. As suggested by the clinical trials [9], an effective IAS therapy highly depends on the individual patient. Thus, we modified the model by taking parametric variation caused by personalized differences into account. In detail, according to clinical data from hundreds of patients [8], we replaced six system parameters with random variables having appropriate (continuous) distributions, including α_k (the proliferation rate of androgen-dependent (AD) cells), α_y (the proliferation rate of androgen-independent (AI) cells), β_k (the apoptosis rate of AD cells), β_y (the apoptosis rate of AI cells), m_1 (the mutation rate from AD to AI cells), and z_0 (the normal androgen level). To describe the variations due to individual differences, we assigned α_k to be U(0.0193, 0.0214), α_y to be U(0.0230, 0.0254), β_k to be U(0.0072, 0.0079), β_y to be U(0.0160, 0.0176), m_1 to be U(0.000475, 0.000525), and z_0 to N(30.0, 0.001). We used SReach to estimate the probabilities of preventing the relapse of prostate cancer with three distinct pairs of treatment thresholds (i.e., combinations of r_0 and r_1). In the experiments, we chose k = 2 as the unfolding depth. For each sample generated, SReach analyzed systems with 41 variables, and 10 ODEs. As shown in Table 2, the model with thresholds r_0 = 10 and r_1 = 15 has a maximum posterior probability that approaches 1, indicating that these thresholds may be considered for the general treatment.

Table 2: Results for the prostate cancer treatment model. #S = number of δ-sat samples, #T = total number of samples, r_0 = lower threshold of the serum PSA level, r_1 = upper threshold, Est_P = estimated maximum posterior probability, A_T(s) = average CPU time of each sample in seconds, and Tot_T(s) = total CPU time for all samples in seconds.

<table>
<thead>
<tr>
<th>(r_0, r_1)</th>
<th>Est_P</th>
<th>#S</th>
<th>#T</th>
<th>A_T(s)</th>
<th>Tot_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(5, 10)</td>
<td>0.996</td>
<td>8226</td>
<td>16584</td>
<td>0.596</td>
<td>9892</td>
</tr>
<tr>
<td>(7, 11)</td>
<td>0.994</td>
<td>335</td>
<td>336</td>
<td>54.307</td>
<td>18247</td>
</tr>
<tr>
<td>(10, 15)</td>
<td>0.996</td>
<td>240</td>
<td>240</td>
<td>50.650</td>
<td>121560</td>
</tr>
</tbody>
</table>

Synthesized Killered Model. Due to the widespread misuse and overuse of antibiotics, drug-resistant bacteria now pose significant risks to health, agriculture, and the environment. An alternative to conventional antibiotics is phage-based therapy. Our approach to antibiotic resistance is to engineer a temperate phage, Lambda (λ), with light-activated production of superoxide (SOX). We incorporated the Killered protein which has been shown to be toxic, and can provide another level of controlled bacteria killing [28]. A probabilistic hybrid automaton for this synthesized Killered model, as shown in Figure 2 in Appendix E, has been constructed. Considering individual differences of bacterial cells and distinct experimental environments, additional randomness on transition probabilities were considered. SReach was first used to validate this model by estimating the probabilities of killing bacterial cells with different values for k, as shown in Table 3. We noticed that the probabilities of paths going through mode 6 to mode 11 in Figure 2 are close to 0. To exclude the effect from sampling of rare events, we increased the probability of entering mode 6. After this modification, the corresponding probabilities estimated by SReach still approach 0. We conclude that it is impossible for this model to enter mode 6. SReach was also used to (a) find out the relation between the time to turn on the light after adding the molecular biology reagent IPTG and the total time to kill bacterial cells (see Table 3 in Appendix D. (b) figure out that the lower bound for the duration of exposure to light is 3 (see Table 3 in Appendix D. (c) find that the time to remove IPTG is not sensitive considering whether bacterial cells will be killed, and (d) estimate that the upper bound of SOX_kill (the necessary concentration of SOX to kill bacterial cells) is 0.6667. All these findings have been reported to biologists for further checking.

Table 3: Results for the killed model. #S = number of δ-sat samples, #T = total number of samples, r_0 = lower threshold of the serum PSA level, r_1 = upper threshold, Est_P = estimated maximum posterior probability, A_T(s) = average CPU time of each sample in seconds, and Tot_T(s) = total CPU time for all samples in seconds.

<table>
<thead>
<tr>
<th>k</th>
<th>Est_P</th>
<th>#S</th>
<th>#T</th>
<th>A_T(s)</th>
<th>Tot_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.544</td>
<td>8954</td>
<td>16452</td>
<td>0.074</td>
<td>1219.38</td>
</tr>
<tr>
<td>6</td>
<td>0.247</td>
<td>3045</td>
<td>12336</td>
<td>0.969</td>
<td>11957.12</td>
</tr>
<tr>
<td>7</td>
<td>0.096</td>
<td>559</td>
<td>5808</td>
<td>5.470</td>
<td>31770.36</td>
</tr>
<tr>
<td>8</td>
<td>0.004</td>
<td>0</td>
<td>240</td>
<td>0.004</td>
<td>0.88</td>
</tr>
<tr>
<td>9</td>
<td>0.004</td>
<td>0</td>
<td>240</td>
<td>0.012</td>
<td>2.97</td>
</tr>
<tr>
<td>10</td>
<td>0.004</td>
<td>0</td>
<td>240</td>
<td>0.013</td>
<td>3.18</td>
</tr>
</tbody>
</table>

5. CONCLUSIONS AND FUTURE WORK

We have presented a probabilistic bounded reachability analysis tool that combines δ-decision procedures and statistical tests. It supports reachability analysis for hybrid systems with parametric uncertainty and probabilistic hybrid automata with additional randomness. It takes the full nondeterminism of models into account, and ensures the estimation accuracy by placing arbitrary small error bounds. This tool has been used for the probabilistic bounded reachability analysis of three representative examples - a prostate cancer treatment model, a cardiac model, and a synthesized Killered model - which are currently out of the reach of other formal tools. In the near future, we plan to extend support for more general stochastic hybrid models that include probabilistic jumps with continuous distributions, and stochastic differential equations.
6. REFERENCES


APPENDIX

A. STATISTICAL TESTS

In this section we briefly describe the statistical techniques implemented in SReach. To deal with qualitative questions, SReach supports the following hypothesis testing methods.

Lai’s test [26]. As a simple class of sequential tests, it tests the one-sided composite hypotheses \( H_0 : \theta \leq \theta_0 \) versus \( H_1 : \theta > \theta_1 \) for the natural parameter \( \theta \) of an exponential family of distributions under the 0–1 loss and cost \( c \) per observation. [26] shows that these tests have nearly optimal frequentist properties and also provide approximate Bayes solutions with respect to a large class of priors.

Bayesian factor test [24]. The use of Bayes factors is a Bayesian alternative to classical hypothesis testing. It is based on the Bayes theorem. Hypothesis testing with Bayes factors is more robust than frequentist hypothesis testing, as the Bayesian form avoids model selection bias, evaluates evidence in favor of the null hypothesis, includes model uncertainty, and allows non-nested models to be compared. Also, frequentist significance tests become biased in favor of rejecting the null hypothesis with sufficiently large sample size.

Bayesian factor test with indifference region. A hypothesis test has ideal performance if the probability of the Type-I error (respectively, Type-II error) is exactly \( \alpha \) (respectively, \( \beta \)). However, these requirements make it impossible to ensure a low probability for both types of errors simultaneously (see [35] for details). A solution is to use an indifference region. The indifference region indicates the distance between two hypotheses, which is set to separate the two hypotheses.

Sequential probability ratio test (SPRT) [34]. The SPRT considers a simple hypothesis \( H_0 : \theta = \theta_0 \) against a simple alternative \( H_1 : \theta = \theta_1 \). With the critical region \( \Lambda_n \) and two thresholds \( A \) and \( B \), SPRT decides that \( H_0 \) is true and stops when \( \Lambda_n > B \). It decides that \( H_1 \) is true and terminates if \( \Lambda_n < A \). If \( A < \Lambda_n < B \), it will collect another observation to obtain a new critical region \( \Lambda_{n+1} \). The SPRT is optimal, among all sequential tests, in the sense that it minimizes the average sample size.

To offer quantitative answers, SReach also supports estimation procedures as below.

Chernoff-Hoeffding bound [22]. To estimate the mean \( p \) of a (bounded) random variable, given a precision \( \delta \) and coverage probability \( \alpha \), the Chernoff-Hoeffding bound computes a value \( p' \) such that \( |p' - p| \leq \delta \) with probability at least \( \alpha \).

Bayesian Interval Estimation with Beta prior [37]. This method estimates \( p \), the unknown probability that a random sampled model satisfies a specified reachability property. The estimate will be in the form of a confidence interval, containing \( p \) with an arbitrary high probability. [37] assumes that the unknown \( p \) is given by a random variable, whose density is called the prior density, and focuses on Beta priors.

Direct sampling. Given \( N \) as the number of samples to be sampled, the direct sampling method estimates the mean of \( p \) of a (bounded) random variable. According to the central limit theorem [14], the error \( \epsilon \) with a confidence \( c \) between the real probability \( p \) and the estimated \( \hat{p} \) is bounded:

\[
\epsilon = \phi^{-1}\left(\frac{c+1}{2}\right) \frac{\phi\left(\frac{1-p}{N}\right)}{\sqrt{\frac{1}{N} + \frac{1}{N} \phi^2\left(\frac{1-p}{N}\right)^2}}
\]

where \( \phi(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} e^{-t^2/2} dt \). That is, as \( N \) goes to \( \infty \), the estimated probability approaches to the real one.

B. \( \delta \)-DECISIONS FOR HYBRID MODELS

The reachability problems of hybrid automata can be encoded using a first-order language \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \) over the reals, which allows the use of a wide range of real functions including nonlinear ODEs. Then, \( \delta \)-complete decision procedures are used to find solutions to these formulas to synthesize parameters.

\( \mathcal{L}_{\mathbb{R},\mathbb{F}} \)-Formulas Let \( \mathcal{F} \) be a collection of computable real functions. We define:

\[
t := x \mid f(t(x)), \text{ where } f \in \mathcal{F} \text{ (constants are 0-ary functions); }
\]

\[
\varphi := t(x) > 0 \mid t(x) \geq 0 \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \exists x_i \varphi \mid \forall x_i \varphi.
\]

By computable real function we mean Type 2 computable, which informally requires that a (real) function can be algorithmically evaluated with arbitrary accuracy. Since in general \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \) formulas are undecidable, the decision problem needs to be relaxed. In particular, for any \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \) formula \( \phi \) and any rational \( \delta > 0 \) one can obtain a \( \delta \)-weakening formula \( \phi^\delta \) from \( \phi \) by substituting the atoms \( t > 0 \) with \( t > -\delta \) (and similarly for \( t \geq 0 \)). Obviously, \( \phi \) implies \( \phi^\delta \), but not the vice versa. Now, the \( \delta \)-decision problem is deciding correctly whether:

- \( \phi \) is false (unsat);
- \( \phi^\delta \) is true (sat).

If both cases are true, then either decision is correct. More details on algorithms (\( \delta \)-complete decision procedures) for solving \( \delta \)-decision problems for \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \) and for ODEs can be found in [17][19].

Now we state the encoding for hybrid models. Hybrid automata generalize finite-state automata by permitting continuous time flow in each discrete mode. Also, in each mode an invariant must be satisfied by the flow, and mode switches are controlled by jump conditions.

\( \mathcal{L}_{\mathbb{R},\mathbb{F}} \)-Representations of Hybrid Automata A hybrid automaton in \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \)-representation is a tuple

\[
H = (X, Q, \{\text{flow}_q(x, y, t) : q \in Q\}, \{\text{inv}_q(x) : q \in Q\},
\]

\[
\{\text{jump}_{q \rightarrow q'}(x, y) : q, q' \in Q\}, \{\text{init}_q(x) : q \in Q\})
\]

where \( X \subseteq \mathbb{R}^n \) for some \( n \in \mathbb{N} \), \( Q = \{q_1, ..., q_m\} \) is a finite set of modes, and the other components are finite sets of quantifier-free \( \mathcal{L}_{\mathbb{R},\mathbb{F}} \)-formulas.

We now show the encoding of bounded reachability, which is used for encoding the parameter synthesis problem. We
want to decide whether a given hybrid system reaches a particular region of its state space after following a (bounded) number of discrete transitions, i.e., jumps. First, we need to define auxiliary formulas used for ensuring that a particular mode is picked at a certain step.

**Definition** Let \( Q = \{q_1, \ldots, q_m\} \) be a set of modes. For any \( q \in Q \), and \( i \in \mathbb{N} \), use \( b_q^i \) to represent a Boolean variable. We now define

\[
\text{enforce}_Q(q, i) = b_q^i \land \bigwedge_{p \in Q(q)} \lnot b_p^i
\]

\[
\text{enforce}_Q(q, q', i) = b_q^i \land \lnot b_{q'}^{i+1} \land \bigwedge_{p \in Q(q)} \lnot b_p^i \land \bigwedge_{p' \in Q(q')} \lnot b_{p'}^{i+1}
\]

We omit the subscript \( Q \) when the context is clear.

We can now define the following formula that checks whether a goal region of the automaton state space is reachable after exactly \( k \) discrete transitions. We first state the simpler case of a hybrid system without invariants.

**\( k \)-Step Reachability, Invariant-Free Case** Suppose \( H \) is an invariant-free hybrid automaton, \( U \) a subset of its state space represented by \( \text{goal} \), and \( M > 0 \). The formula \( \text{Reach}_{H,U}(k, M) \) is defined as:

\[
\exists^k \vec{x}_0 \exists^k \vec{x}_0 \cdots \exists^k \vec{x}_M \exists^k \vec{x}_M \exists^{[0, M]} t_0 \cdots \exists^{[0, M]} t_k.
\]

\[
\bigvee_{q \in Q} \left( \text{init}_q(\vec{x}_0) \land \text{flow}_q(\vec{x}_0, \vec{x}_0, t_0) \land \text{enforce}(q, 0) \right)
\]

\[\land \bigwedge_{i=0}^{k-1} \left( \bigvee_{q, q' \in Q} \left( \text{jump}_{q \rightarrow q'}(\vec{x}_i, \vec{x}_{i+1}) \land \text{enforce}(q, q', i) \right) \land \text{flow}_{q'}(\vec{x}_{i+1}, \vec{x}_{i+1}, t_{i+1}) \land \text{enforce}(q', i+1) \right) \land \bigwedge_{q \in Q} \left( \text{goal}_q(\vec{x}_k) \land \text{enforce}(q, k) \right)
\]

where \( \exists^k x \) is a shorthand for \( \exists x \) for all the time points along a continuous flow, the invariant variables \( \vec{x} \) must take values that satisfy the invariant conditions \text{inv}_q(\vec{x})

The extra universal quantifier for each continuous flow expresses the requirement that for all the time points between the initial and ending time point \( t \in [0, t_i + 1] \) in a flow, the continuous variables \( \vec{x} \) must take values that satisfy the invariant conditions \text{inv}_q(\vec{x})

**C. THE SREACH TOOL**

**C.1 Input format**

The inputs to our SReach tool are descriptions of (probabilistic) hybrid automata with random variables (representing the probabilistic system parameters, and probabilistic jumps), and the reachability property to be checked. Following roughly the same format as the above definition of (probabilistic) hybrid automata, and adding the declarations of random variables, the description of an automaton is as follows.

**Preprocessor.** We can use the C language syntax to define constants and macros.

**Variable declaration.** For a random variable, the declaration specifies its distribution and name. Variables which are not random variables are required to be declared within bounds.

**(Probabilistic) Hybrid automaton.** A (probabilistic) hybrid automaton is represented by a set of modes. Within each mode declaration, we can specify statements for the mode invariant(s), flow function(s), and (probabilistic) jump condition(s). For a mode invariant, we can give any logic formula of the variables. A flow function is expressed by an ODE. As for a nonprobabilistic jump condition, it is written as:

\[<\text{logic_formula1}> \implies 0\text{@target_mode} <\text{logic_formula2}>\]

where the first logic formula is given as the guard of the jump, and the second one specifies the reset condition after the jump. While for a probabilistic jump condition, we need an extra constraint to express the stochastic choice, which is of the following form:

\[\text{and} <\text{logic_formula1}> <\text{stochastic_choice}> \implies 0\text{@target_mode} <\text{logic_formula2}>\]
where the stochastic choice is a formula indicating which
probabilistic transition will be chosen for this jump.

**Initial conditions and Goals.** Following the declaration
of modes, we can declare one initial mode with correspond-
ing conditions, and the reachability properties in the end.

**Example 1.** The following is an example input file for a
hybrid automaton with parametric uncertainty. Currently,
users can specify random variables (representing certain sys-
tem parameters) with Bernoulli distribution (B), Uniform
distribution (U), Gaussian distribution (N), Exponential dis-
tribution (E), and general Discrete distribution with given
possible values and corresponding probabilities (DD).

```plaintext
#define pi 3.1416
N(1.0,1) mu1;
U(10,15) thro;
E(0.49) theta1;
DD(0.07, 1:0.3) mu2;
[0,5] x;
[0,3] time;
{ mode 1;
  invt:
    (x<=1.5);
    (x>=0);
  flow:
    d/dt[x] = thro*(1/(theta1*sqrt(2*pi)))*exp(-((x-mu1+mu2)^2)/(2*theta1^2));
  jump:
    (x>(theta1+5)) ==> 02(x'=x);
} init:
@1 (x=xinit);
goal:
@4 (x>=50);
```

**Example 2.** This example demonstrates the format of the in-
put file for a probabilistic hybrid automaton with additional
randomness for transition probabilities. Note that, unlike
the notations of declarations of random variables represent-
ing system parameters and probabilistic transitions, decla-
rations of random variables used to express the additional
randomness for jump probabilities start with a prefix
`j`.

```plaintext
jU(0.7, 0.9) pjumprv;
DD(1:pjumprv, 2:1-pjumprv) pjump1;
DD(1:0.3, 2:0.7) pjump2;
[-1000, 1000] x;
[-1000, 1000] y;
[0,3] time;
{ mode 1;
  invt:
    (x <= 2);
    (x > 0);
    (y <= 7.7);
    (y > -3);
  flow:
    d/dt[x] = x * y;
    d/dt[y] = 3 * x - y;
  jump:
    (and (abs(y)*x - 2 <= x / 2) (pjump1 = 1)) ==> @1 (and (x' > sin(y)) (y' <= 4 * y));
    (and (abs(y)*x - 2 <= x / 2) (pjump1 = 2)) ==> @2 (and (x' <= 3.1) (y' = 2 * x));
}
```

**C.2 Command line**

*SReach* offers two choices. It can be run sequentially by typing

```
sreach_sq <statistical_testing_option> <filename>
dReach <k> <delta>,
```

or in parallel by

```
sreach_para <statistical_testing_option> <filename>
dReach <k> <delta>,
```

where:

- **statistical_testing_option** is a text file containing
  a sequence of test specifications. We will introduce
  the usages of statistical testing options in the follow-
  ing part;
- **filename** is a .pdrh file describing the model of a hybrid
  system with probabilistic system parameters. It is of
  the input format described in last sub-section;
- **dReach** is a tool for bounded reachability analysis of
  hybrid systems based on dReal;
- **k** is the number of steps of the model that the tool will
  explore; and
- **delta** is the precision for the δ-decision problem.

**C.3 Statistical testing options**

*SReach* can be used with different statistical testing methods
through the following specifications.

- **Lai’s test**: Lai <theta> <cost_per_sample>, where `theta`
  indicates the probability threshold.

- **Bayes factor test**: BFT <theta> <T> <alpha> <beta>, where
  `theta` is a probability threshold satisfying \(0 < \theta < 1\),
  `T` is a ratio threshold satisfying \(T > 1\), and `alpha`
  and `beta` are beta prior parameters.
CHB <delta> <coverage_probability> Chernoff-Hoeffding bound: CHB <delta> <coverage_probability>, where <delta> is the given precision, and coverage probability indicates the confidence.

Bayesian Interval Estimation with Beta prior: BEST <delta> <coverage_probability> <alpha> <beta>.

Direct/Naïve Sampling: NSAM <num_of_samples>.

D. MODEL DESCRIPTION

Atrial Fibrillation. The model has four discrete control locations, four state variables, and nonlinear ODEs. A typical set of ODEs in the model is:

\[
\begin{align*}
\frac{du}{dt} & = e + (u - \theta_v)(u - u)v_{f1} + 
\frac{g_{\omega_2}}{1 + \exp(-2k(u - u_s))} - g_{\omega_2} s \\
\frac{ds}{dt} & = -g_{\omega_2} v \\
\frac{dv}{dt} & = -g_{\omega_2} v \\
\frac{d\omega_2}{dt} & = -c_3 \cdot \omega_1 \omega_2 = -c_4 \cdot \omega_2 \\
\end{align*}
\]

The exponential term on the right-hand side of the ODE is the sigmoid function, which often appears in modelling biological switches.

Electronic Oscillator. The 3dOsc model represents an electronic oscillator model that contains nonlinear ODEs such as the following:

\[
\begin{align*}
\frac{dx}{dt} & = -ax \cdot \sin(\omega_1 \cdot \tau) \\
\frac{dy}{dt} & = -ay \cdot \sin((\omega_1 + c_1) \cdot \tau) \cdot \sin(\omega_2) \cdot 2 \\
\frac{dz}{dt} & = -az \cdot \sin((\omega_2 + c_2) \cdot \tau) \cdot \cos(\omega_1) \cdot 2 \\
\frac{\omega_1}{dt} & = -c_3 \cdot \omega_1 \omega_2 = -c_4 \cdot \omega_2 \\
\end{align*}
\]

Quadcopter Control. We developed a model that contains the full dynamics of a quadcopter. We use the model to solve control problems by answering reachability questions.

\[
\begin{align*}
\frac{dx}{dt} & = (\alpha_x(k_1 + (1 - k_1)z^2 + k_2) - \beta_x((1 - k_3)z + k_5) + k_5) \\
\frac{dy}{dt} & = m_1 (1 - z) x + (\alpha_y(1 - d - z^2) - \beta_y) y + c_2 y \\
\frac{dz}{dt} & = -z + c_1 z \\
\frac{dv}{dt} & = (\alpha_x(k_1 + (1 - k_1)z + k_2) - \beta_x((1 - k_3)z + k_5) + k_5) \\
& - m_1 (1 - z) x + c_1 x + m_1 (1 - z) x + (\alpha_y(1 - d - z^2) - \beta_y) y + c_2 y \\
\end{align*}
\]

Prostate Cancer Treatment. The Prostate Cancer Treatment model exhibits more nonlinear ODEs.

\[
\begin{align*}
\frac{dx}{dt} & = (\alpha_x(k_1 + (1 - k_1)z^2 + k_2) - \beta_x((1 - k_3)z^2 + k_5) + k_5) \\
& - m_1 (1 - z) x + c_1 x \\
\frac{dy}{dt} & = m_1 (1 - z) x + (\alpha_y(1 - d - z^2) - \beta_y) y + c_2 y \\
\frac{dz}{dt} & = -z + c_1 z \\
\frac{dv}{dt} & = (\alpha_x(k_1 + (1 - k_1)z^2 + k_2) - \beta_x((1 - k_3)z^2 + k_5) + k_5) \\
& - m_1 (1 - z) x + c_1 x + m_1 (1 - z) x + (\alpha_y(1 - d - z^2) - \beta_y) y + c_2 y \\
\end{align*}
\]

Synthesized Killerred Model. The ODEs missing in Fig-
Figure 2: A probabilistic hybrid automaton for synthesized phage-based therapy model

The table 4 shows the results of experiments. To further demonstrate the feasibility of SReach, these experiments were conducted with the sequential version of SReach on a machine with 2.9GHz Intel Core i7 processor and 8GB RAM, running OS X 10.9.2. In our experiments, we used \( \alpha = 0.001 \) as the precision for the \( \delta \)-decision problem and Bayesian sequential estimation with 0.01 half-interval width, coverage probability 0.99, and uniform prior \( (\alpha = \beta = 1) \). In the following table, BB refers to the bouncing ball models, Tld the thermostat model with linear temperature decrease, Ted the thermostat model with exponential decrease, DT the dual thermostat models, W the watertank models, DW the dual watertank models, Que the model for queuing system which
### Table 4: #Ms = number of modes, K indicates the unfolding steps, #ODEs = number of ODEs in the model, #Vs = number of total variables in the unfolded formulae, #RVs = number of random variables in the model, $\delta$ = precision used in $dReach$, #S = number of $\delta$-sat samples, #T = total number of samples, Est_P = estimated maximum posterior probability, A_T(s) = average CPU time of each sample in seconds, and T_T(s) = total CPU time for all samples in seconds.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Ms</th>
<th>K</th>
<th>#ODEs</th>
<th>#Vs</th>
<th>#RVs</th>
<th>$\delta$</th>
<th>Est_P</th>
<th>#S</th>
<th>#T</th>
<th>A_T(s)</th>
<th>T_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>BBK1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>14</td>
<td>3</td>
<td>0.001</td>
<td>0.754</td>
<td>5372</td>
<td>7126</td>
<td>0.086</td>
<td>612.836</td>
</tr>
<tr>
<td>BBK5</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>38</td>
<td>3</td>
<td>0.001</td>
<td>0.059</td>
<td>209</td>
<td>3628</td>
<td>0.253</td>
<td>917.884</td>
</tr>
<tr>
<td>BBwDv1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>3</td>
<td>0.001</td>
<td>0.208</td>
<td>2206</td>
<td>10919</td>
<td>0.080</td>
<td>873.522</td>
</tr>
<tr>
<td>BBwDv2K2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>3</td>
<td>0.001</td>
<td>0.845</td>
<td>7330</td>
<td>8669</td>
<td>0.209</td>
<td>1811.821</td>
</tr>
<tr>
<td>BBwDv2K8</td>
<td>2</td>
<td>8</td>
<td>4</td>
<td>56</td>
<td>3</td>
<td>0.001</td>
<td>0.207</td>
<td>2259</td>
<td>10901</td>
<td>0.858</td>
<td>9353.058</td>
</tr>
<tr>
<td>Tld</td>
<td>2</td>
<td>7</td>
<td>2</td>
<td>33</td>
<td>4</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>0.213</td>
<td>48.351</td>
</tr>
<tr>
<td>Ted</td>
<td>2</td>
<td>7</td>
<td>4</td>
<td>50</td>
<td>4</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>0.382</td>
<td>110.931</td>
</tr>
<tr>
<td>DTldK3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>38</td>
<td>2</td>
<td>0.001</td>
<td>0.161</td>
<td>1442</td>
<td>8961</td>
<td>0.280</td>
<td>2509.078</td>
</tr>
<tr>
<td>DTldK5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>38</td>
<td>2</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>0.215</td>
<td>48.806</td>
</tr>
<tr>
<td>W4mv1</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>26</td>
<td>6</td>
<td>0.001</td>
<td>0.161</td>
<td>1442</td>
<td>8961</td>
<td>0.247</td>
<td>27.240</td>
</tr>
<tr>
<td>W4mv2K3</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>26</td>
<td>6</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>0.673</td>
<td>152.771</td>
</tr>
<tr>
<td>W4mv2K7</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>50</td>
<td>6</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>2.15</td>
<td>48.806</td>
</tr>
<tr>
<td>DWK1</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>14</td>
<td>5</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>5.144</td>
<td>1167.688</td>
</tr>
<tr>
<td>DWK3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>26</td>
<td>5</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>5.144</td>
<td>1167.688</td>
</tr>
<tr>
<td>DWK9</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>62</td>
<td>5</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>5.144</td>
<td>1167.688</td>
</tr>
<tr>
<td>Que</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>13</td>
<td>4</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>8.273</td>
<td>1877.969</td>
</tr>
<tr>
<td>3dOsc</td>
<td>3</td>
<td>2</td>
<td>18</td>
<td>48</td>
<td>2</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>8.273</td>
<td>1877.969</td>
</tr>
<tr>
<td>QuadC</td>
<td>1</td>
<td>10</td>
<td>14</td>
<td>44</td>
<td>6</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>8.273</td>
<td>1877.969</td>
</tr>
<tr>
<td>ExPHA01</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>2</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>8.273</td>
<td>1877.969</td>
</tr>
<tr>
<td>ExPHA02</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>17</td>
<td>1</td>
<td>0.001</td>
<td>0.900</td>
<td>5361</td>
<td>5953</td>
<td>0.0004</td>
<td>3.35</td>
</tr>
<tr>
<td>KRk5</td>
<td>6</td>
<td>5</td>
<td>84</td>
<td>194</td>
<td>2</td>
<td>0.001</td>
<td>0.544</td>
<td>8946</td>
<td>16457</td>
<td>0.122</td>
<td>2015.64</td>
</tr>
<tr>
<td>KRk6</td>
<td>8</td>
<td>6</td>
<td>112</td>
<td>224</td>
<td>6</td>
<td>0.001</td>
<td>0.246</td>
<td>2032</td>
<td>8263</td>
<td>1.385</td>
<td>11444.22</td>
</tr>
<tr>
<td>KRk7</td>
<td>10</td>
<td>7</td>
<td>150</td>
<td>271</td>
<td>6</td>
<td>0.001</td>
<td>0.996</td>
<td>558</td>
<td>5795</td>
<td>16.275</td>
<td>94311.18</td>
</tr>
<tr>
<td>KRk8</td>
<td>7</td>
<td>8</td>
<td>105</td>
<td>303</td>
<td>6</td>
<td>0.001</td>
<td>0.004</td>
<td>227</td>
<td>0.003</td>
<td>0.58</td>
<td></td>
</tr>
<tr>
<td>KRk9</td>
<td>9</td>
<td>9</td>
<td>135</td>
<td>335</td>
<td>6</td>
<td>0.001</td>
<td>0.004</td>
<td>227</td>
<td>227</td>
<td>0.015</td>
<td>3.43</td>
</tr>
<tr>
<td>KRk10</td>
<td>11</td>
<td>10</td>
<td>165</td>
<td>367</td>
<td>6</td>
<td>0.001</td>
<td>0.004</td>
<td>227</td>
<td>227</td>
<td>0.026</td>
<td>5.92</td>
</tr>
</tbody>
</table>

Table 5: The relation between the time to turn on the light after adding IPTG and the total time to kill bacteria cells ($k = 5$).

<table>
<thead>
<tr>
<th>$t_{\text{light on}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{\text{tot}}$</td>
<td>16</td>
<td>17.2</td>
<td>18.5</td>
<td>20</td>
<td>21.3</td>
<td>22.7</td>
<td>23.5</td>
<td>24.1</td>
<td>25</td>
<td>30</td>
</tr>
</tbody>
</table>

Table 6: The impact of the time duration that the cells are exposed to light ($k = 6$).

<table>
<thead>
<tr>
<th>$t_{\text{light off}}$</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>killbacteriaceells</td>
<td>Failed</td>
<td>Failed</td>
<td>Failed</td>
<td>Succ</td>
<td>Succ</td>
<td>Succ</td>
<td>Succ</td>
<td>Succ</td>
<td>Succ</td>
<td>Succ</td>
</tr>
</tbody>
</table>

has both nonlinear functions and nondeterministic jumps, 3dOsc the model for 3d oscillator, and QuadC the model for quadcopter stabilization control. Following these hybrid systems with parametric uncertainty, we also consider two example PHAs - ExPHA01 and ExPHA02, and PHAs with additional randomness - KR our killerred models.