SReach: A Probabilistic Bounded $\delta$-Reachability Analyzer for Stochastic Hybrid Systems

Qinsi Wang$^1$, Paolo Zuliani$^2$, Soonho Kong$^3$, Sicun Gao$^3$, Edmund M. Clarke$^1$

$^1$ Computer Science Department, Carnegie Mellon University, USA  
$^2$ School of Computing Science, Newcastle University, UK  
$^3$ CSAIL, Massachusetts Institute of Technology, USA

Abstract. In this paper, we present a new tool $SReach$, which solves probabilistic bounded reachability problems for two classes of models 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. $SReach$ encodes stochastic information by using a set of introduced random variables, and combines $\delta$-complete decision procedures and statistical tests to solve $\delta$-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. Compared to standard simulation-based methods, it supports non-deterministic branching, increases the coverage of simulation, and avoids the zero-crossing problem. We demonstrate $SReach$’s applicability by discussing three representative biological models and 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 the generality, they have been widely used in various areas, including biological systems, financial decision problems, and cyber-physical systems [2, 6]. One elementary question for the quantitative analysis of SHSs is the probabilistic reachability problem, considering that many verification problems can be reduced to reachability problems. It is to compute the probability of reaching a certain set of states. The set may represent certain unsafe states which should be avoided or visited only with some small probability, or dually, good states which should be visited frequently. This problem is no longer a decision problem, as it generalizes that by asking what is the probability that the system reaches the target region. For SHSs with both stochastic and non-deterministic behavior, the problem results in general in a range of probabilities, thereby becoming an optimization problem.

* This research was sponsored by the Air Force Office of Scientific Research (FA9550-12-1-0146) and the Office of Naval Research (N000141310090).
To describe stochastic dynamics, uncertainties have been added to hybrid systems in various ways. One way expresses random initial values and stochastic dynamical coefficients using random variables, resulting in hybrid automata (HAs) [16] 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) [23]. When continuous probabilistic events are also involved, we call them stochastic hybrid automata (SHAs) [10]. Other models substitute deterministic flows with stochastic ones, such as stochastic differential equations (SDEs) [1], where the random perturbation affects the dynamics continuously. When all such modifications have been applied, the resulting models are called general stochastic hybrid systems (GSHSs) [18]. Among these different models, of particular interest for this paper are HAs with parametric uncertainty and PHAs with additional randomness for both transition probabilities and variable resets. Note that, in the following, we use notations - HA_p and PHA_r - for these two model classes respectively.

When modeling real-world systems, such as biological systems and cyber-physical 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 HA_p’s are suitable models considering these major causes. Note that, in both cases, we assume that the probability distributions of probabilistic system parameters are known and remain unchanged throughout the systems evolution.

As another interesting and more expressive class of models, PHAs extend HAs with discrete probability distributions. More precisely, for discrete transitions in a model, instead of making a purely (non)deterministic choice over the set of currently enabled jumps, a PHA (non)deterministically chooses among the set of recently enabled discrete probability distributions, each of which is defined over a set of transitions. Although randomness only influences the discrete dynamics of the model, PHAs are still very useful and have interesting practical applications [24]. In this paper, we consider a variation of PHAs, where additional randomness for both transition probabilities and resets of system variables are allowed. In other words, in terms of the additional 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 a fixed value.

In this paper, we describe our tool \textit{SReach} which supports probabilistic
bounded \( \delta \)-reachability analysis for the above two model classes. It combines
the recently proposed \( \delta \)-complete bounded reachability analysis technique [12]
with statistical testing techniques. \textit{SReach} saves the virtues of the Satisfiability
Modulo Theories (SMT) based Bounded Model Checking (BMC) for HAs [7,26],
namely the fully symbolic treatment of hybrid state spaces, while advancing
the reasoning power to probabilistic models. Furthermore, by utilizing the \( \delta \)-
complete analysis method, the full non-determinism of models will be considered.
The coverage of simulation will be increased, as the \( \delta \)-complete analysis method
results in an over-approximation of the reachable set, whereas simulation is only
an under-approximation of it. The zero-crossing problem can be avoided as, if a
zero-crossing point exists, it will always return an interval containing it. By using
statistical tests, \textit{SReach} can place controllable error bounds on the estimated
probabilities. We discuss three biological models - an atrial fibrillation model, a
prostate cancer treatment model, and our synthesized Killerred biological model
- to show that \textit{SReach} can answer questions including model validation/falsifi-
cation, parameter synthesis, and sensitivity analysis. To further demonstrate
its applicability, we also apply it to additional real-world hybrid systems with
parametric uncertainty.

\textbf{Related work.} Hahn et al. promoted an abstraction-based method where
the given PHA is abstracted into an \( n \)-player stochastic game [15], albeit being
limited to linear dynamics. Fränzle at al. proposed a Stochastic SMT-based
Ellen et al. [9] proposed a statistical model checking technique for verifying hy-
brid systems with continuous non-determinism, thereby expanding the class of
systems analyzable, yet confined dynamics to (non-linear) pre-post conditions
rather than ODEs. \textit{SReach} supports both discrete and continuous random vari-
ables, and ODEs. ProbReach [22] also uses the \( \delta \)-complete procedures and offers
verified estimated probability interval containing the real probability, yet can
only deal with hybrid systems with initial random variables. While \textit{SReach} is
able to handle probabilistic transitions as well.

The paper proceeds by introducing two model classes of SHSs under consider-
atation in Section 2, Section 3 formally states probabilistic bounded \( \delta \)-reachability
problems and explains how \textit{SReach} solves these problems by combining \( \delta \)-complete
decision procedures with statistical tests. Case studies and additional experi-
ments are discussed in Section 4. Section 5 concludes the paper.

2 Stochastic Hybrid Models

Before introducing the algorithm implemented by \textit{SReach} and the problems that
it can handle, we first define two model classes that \textit{SReach} considers formally.
For \( HA_p \)'s, we follow the definition of HAs in [16], and extend it to consider
probabilistic parameters in the following way.
Definition 1 (HA_p). A hybrid automaton with parametric uncertainty is a tuple $H_p = \langle (Q, E), V, RV, \text{Init}, \text{Flow}, \text{Inv}, \text{Jump}, \Sigma \rangle$, where

- 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. We write $\dot{V}$ to represent the first derivatives of variables during the continuous change, and write $V'$ to denote values of variables at the conclusion of the discrete change.
- $RV = \{w_1, \cdots, w_k\}$ is a finite set of independent random variables, where the distribution of $w_i$ is denoted by $P_i$.
- Init, Flow, and Inv are labeling functions over $Q$. For each mode $q \in Q$, the initial condition $\text{Init}(q)$ and invariant condition $\text{Inv}(q)$ are predicates whose free variables are from $V \cup RV$, and the flow condition $\text{Flow}(q)$ is a predicate whose free variables are from $V \cup \dot{V} \cup RV$.
- Jump is a transition labeling function that assigns to each transition $e \in E$ a predicate whose free variables are from $V \cup V' \cup RV$.
- $\Sigma$ is a finite set of events, and an edge labeling function $\text{event} : E \to \Sigma$ assigns to each control switch an event.

Another class is PHA_r,s, which extend HAs with discrete probability transitions and additional randomness for transition probabilities and variable resets.

Definition 2 (PHA_r.). A probabilistic hybrid automaton with additional randomness $H_r$ consists of $Q, E, V, RV, \text{Init}, \text{Flow}, \text{Inv}, \Sigma$ as in Definition 1, and $\text{Cmds}$, which is a finite set of probabilistic guarded commands of the form:

$$g \rightarrow p_1 : u_1 + \cdots + p_m : u_m,$$

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 $x \geq 5 \rightarrow p_1 : (x' = \sin(x)) + (1 - p_1) : (x' = p_x)$, where $x$ is a system variable, $p_1$ has a Uniform distribution $U(0.2, 0.9)$, and $p_x$ 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 used, so that no dependence between the current probabilistic jump and previous transitions will be introduced.
3 SReach algorithm

A recently proposed \( \delta \)-complete decision procedure [12] relaxes the reachability problem for HAs in a sound manner: it verifies a conservative approximation of the system behavior, so that bugs will always be detected. The overapproximation can be tight (tunable by an arbitrarily small rational parameter \( \delta \)), and a false alarm with a small \( \delta \) may indicate that the system is fragile, thereby providing valuable information to the system designer (see Appendix A for details). We now define the probabilistic bounded \( \delta \)-reachability problem based on the bounded \( \delta \)-reachability problem defined in [12].

**Definition 3.** The probabilistic bounded \( k \) step \( \delta \)-reachability for a HA\(_p\) \( H_p \) is to compute the probability that \( H_p \) reaches the target region \( T \) in \( k \) steps. Given the set of independent random variables \( r \), \( Pr(r) \) a probability measure over \( r \), and \( \Omega \) the sample space of \( r \), the reachability probability is \( \int_\Omega I_T(r) dPr(r) \), where \( I_T(r) \) is the indicator function which is 1 if \( H_p \) with \( r \) reaches \( T \) in \( k \) steps.

**Definition 4.** For a PHA\(_r\) \( H_r \), the probabilistic bounded \( k \) step \( \delta \)-reachability estimated by SReach is the maximal probability that \( H_r \) reaches the target region \( T \) in \( k \) steps: \( \max_{\sigma \in E} Pr^k_{H_r,\sigma,T}(i) \), where \( E \) is the set of possible executions of \( H \) starting from the initial state \( i \), and \( \sigma \) is an execution in the set \( E \).

**Algorithm 1 SReach**

```plaintext
1: function SREACH(MP, ST, \delta, k)  
2: if MP is a HA\(_p\) then  
3: MP ← EncRM\(_1\)(MP) \( \triangleright \) encode uncertain system parameters  
4: else \( \triangleright \) otherwise a PHA\(_r\)  
5: MP ← EncRM\(_2\)(MP) \( \triangleright \) encode probabilistic jumps and extra randomness  
6: end if  
7: Succ, N ← 0 \( \triangleright \) number of \( \delta \)-sat samples and total samples  
8: Assgn ← \( \emptyset \) \( \triangleright \) record unique sampling assignments and dReach results  
9: RV ← ExtractRV(MP) \( \triangleright \) get the RVs from the probabilistic model  
10: repeat in parallel  
11: \( S_i \) ← Sim(RV) \( \triangleright \) sample the parameters  
12: if \( S_i \) ∈ Assgn.sample then  
13: Res ← Assgn\(.\)(S\(_i\))\_.res \( \triangleright \) no need to call dReach  
14: else  
15: \( M_i \) ← Gen(MP, \( S_i \)) \( \triangleright \) generate a dReach model  
16: Res ← dReach(M\(_i\), \( \delta \), \( k \)) \( \triangleright \) call dReach to solve \( k \)-step \( \delta \)-reachability  
17: end if  
18: if Res = \( \delta \)-sat then Succ ← Succ + 1  
19: end if  
20: N ← N + 1  
21: until ST.done(Succ, N) \( \triangleright \) perform statistical test  
22: return ST.output  
23: end function
```

After encoding uncertainties using random variables, SReach samples them according to the given distributions. For each sample, a corresponding intermediate HA is generated by replacing random variables with their assigned values.
Then, the \( \delta \)-complete analyzer \( dReach \) is utilized to analyze each intermediate
HA \( M_i \), together with the desired precision \( \delta \) and unfolding depth \( k \). The analyzer
returns either unsat or \( \delta \)-sat for \( M_i \). This information is then used by a chosen
statistical testing procedure to decide whether to stop or to repeat the proce-
dure, and to return the estimated probability. The full procedure is illustrated
in Algorithm 1, where \( MP \) is a given stochastic model, and \( ST \) indicates which
statistical testing method will be used (See Appendix B for various statistical
tests that supported by \( SReach \) and the way to control the induced statistical
error bounds). \( Succ \) and \( N \) are used to record the number of \( \delta \)-sat instances and
total samples generated so far respectively, and are then the inputs of \( ST \). Note
that, for a \( PHA_r \), sampling and fixing the choices of all the probabilistic tran-
sitions in advance results in an over-approximation of the original \( PHA_r \), where
safety properties are preserved. To promise a tight over-approximation and cor-
rectness of estimated probabilities, \( SReach \) supports \( PHA_r \)'s with no or subtle
non-determinism. That is, in order to offer a reasonable estimation, for \( PHA_r \),
\( SReach \) is supposed to be used on models with no or few non-deterministic transi-
tions, or where dynamic interleaving between non-deterministic and probabilistic
choices are not important, such as our KillerRed biological model. To improve the
performance of \( SReach \), each sampled assignment and its corresponding \( dReach \)
result are recorded for avoiding redundant calls to \( dReach \). This significantly re-
duces the total calls for \( PHA_r \)'s, as the size of the sample space involving random
variables describing probabilistic jumps is comparatively small. For the example
PHA (as shown in Figure 1), with this heuristic, the total checking time has been
decreased from 11291.31s for 658 samples (17.16s per sample) to 3295.82s (5.01s
per sample). Furthermore, a parallel version of \( SReach \) has been implemented
using OpenMP, where multiple samples and corresponding HAs 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).

\[
\begin{align*}
\text{Mode 1:} & \quad \frac{dy}{dt}[x] = x \cdot y; \\
& \quad \frac{dy}{dt}[y] = 3 \cdot x - y; \\
& \quad \text{inv:} \\
& \quad \begin{cases}
(x < 2); \\
(x \geq 0); \\
(y < 7.7); \\
(y \geq -3);
\end{cases}
\end{align*}
\]

\[
\begin{align*}
\text{Mode 2:} & \quad \frac{dy}{dt}[x] = x; \\
& \quad \frac{dy}{dt}[y] = 3 \cdot x - y^2; \\
& \quad \text{inv:} \\
& \quad \begin{cases}
(x = x); \\
(x' = x); \\
(y' = y); \\
(y' = 4 \cdot y);
\end{cases}
\end{align*}
\]

Fig. 1: An example probabilistic hybrid automaton

Currently, \( SReach \) supports a number of hypothesis testing and statistical esti-
mation techniques including: Lai's test \[20\], Bayes factor test \[19\], Bayes factor
test with indifference region \[28\], Sequential probability ratio test (SPRT) \[27\],
Chernoff-Hoeffding bound \[17\], Bayesian Interval Estimation with Beta prior...
and Direct Sampling. All methods produce answers that are correct up to a precision that can be set arbitrarily by the user. See Appendix B for more details about these statistical testing techniques. 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 probabilistic bounded reachability problems. The model validation/falsification problem with prior knowledge can be encoded as a probabilistic 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 with a desirable probability? If none exists, the model is incorrect regarding the given prior knowledge. The parameter synthesis problem can also be encoded as a probabilistic $k$-step reachability problem. Does there exist a parameter combination for which the model reaches the given goal region in $k$ steps with a desirable probability? 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 probabilistic 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

Both sequential and parallel versions of SReach are available on https://github.com/dreal/SReach (see Appendix C for its usage). Experiments for the following three biological models were conducted on a server with 2* AMD Opteron(tm) Processor 6172 and 32GB RAM (12 cores were used), running on Ubuntu 14.04.1 LTS. In our experiments we used 0.001 as the precision for the $\delta$-decision problem, and Bayesian sequential estimation with 0.01 as the estimation error bound, coverage probability 0.99, and a uniform prior ($\alpha = \beta = 1$). All the details (including discrete modes, continuous dynamics that described by ODEs, nondeterminism, and stochasticity) of models in the following case studies and additional benchmarks can be found on the tool website.

Atrial Fibrillation. The minimum resistor model reproduces experimentally measured characteristics of human ventricular cell dynamics [5]. It 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 synthesize parameters for this stochastic model, i.e., identifying appropriate ranges and distributions for model parame-
We chose two system parameters - $EPI_{TO1}$ and $EPI_{TO2}$, and varied their distributions to see which ones allow the model to present the desired patterns. As in Table 1, when $EPI_{TO1}$ is either close to 400, or between 0.0061 and 0.007, and $EPI_{TO2}$ is close to 6, the model can satisfy the given bounded reachability property with a probability very close to 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>#RVs</th>
<th>$EPI_{TO1}$</th>
<th>$EPI_{TO2}$</th>
<th>#S</th>
<th>S</th>
<th>#T</th>
<th>S</th>
<th>Est_P</th>
<th>T(s)</th>
<th>T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cd_to1_s</td>
<td>1</td>
<td>U(6.1e-3, 7e-3)</td>
<td>6</td>
<td>240</td>
<td>240</td>
<td>0.996</td>
<td>0.270</td>
<td>64.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd_to1_sms</td>
<td>1</td>
<td>U(5.5e-3, 5.9e-3)</td>
<td>6</td>
<td>0</td>
<td>240</td>
<td>0.004</td>
<td>0.042</td>
<td>10.08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd_to2_s</td>
<td>1</td>
<td>400</td>
<td>U(0.131, 6)</td>
<td>240</td>
<td>240</td>
<td>0.996</td>
<td>0.231</td>
<td>55.36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd_to2_sms</td>
<td>1</td>
<td>400</td>
<td>U(0.1, 0.129)</td>
<td>0</td>
<td>240</td>
<td>0.004</td>
<td>0.038</td>
<td>9.15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd_to12_s</td>
<td>2</td>
<td>N(400, 1e-4)</td>
<td>N(6.1e-3, 7e-3)</td>
<td>6</td>
<td>240</td>
<td>0.996</td>
<td>0.270</td>
<td>64.80</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cd_to12_sms</td>
<td>2</td>
<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>
<td>0.037</td>
<td>8.90</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results for the 4-mode atrial fibrillation model ($k = 3$). For each sample generated, $SReach$ analyzed systems with 62 variables and 24 ODEs in the unfolded SMT formulae. #RVs = number of random variables in the model, #S = number of δ-sat samples, #T = total number of samples, Est_P = estimated probability of property, $A_T(s)$ = average CPU time of each sample in seconds, and $T_T(s)$ = total CPU time for all samples in seconds. Note that, we use the same notations in the remaining tables.

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 [25] by adding parametric uncertainty. The IAS therapy switches between treatment-on, and treatment-off with respect to the serum level thresholds of prostate-specific antigen (PSA), namely $r_0$ and $r_1$. As suggested by the clinical trials [4], 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 [3], we replaced six system parameters with random variables having appropriate (continuous) distributions, including $\alpha_x$ (the proliferation rate of androgen-dependent (AD) cells), $\alpha_y$ (the proliferation rate of androgen-independent (AI) cells), $\beta_x$ (the apoptosis rate of AD cells), $\beta_y$ (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 $\alpha_x$ to be $U(0.0193, 0.0214)$, $\alpha_y$ to be $U(0.0230, 0.0254)$, $\beta_x$ to be $U(0.0072, 0.0079)$, $\beta_y$ to be $U(0.0160, 0.0176)$, $m_1$ to be $U(0.0000475, 0.0000525)$, and $z_0$ to be $N(30.0, 0.001)$. 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 $r_1$). 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>
<thead>
<tr>
<th>Model</th>
<th>#RVs</th>
<th>$r_0$</th>
<th>$r_1$</th>
<th>Est_P</th>
<th>#S</th>
<th>#T</th>
<th>S</th>
<th>A_T(s)</th>
<th>T_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCT1</td>
<td>6</td>
<td>5.0</td>
<td>10.0</td>
<td>0.496</td>
<td>8226</td>
<td>16584</td>
<td>0.596</td>
<td>9892</td>
<td></td>
</tr>
<tr>
<td>PCT2</td>
<td>6</td>
<td>7.0</td>
<td>11.0</td>
<td>0.994</td>
<td>345</td>
<td>336</td>
<td>54.307</td>
<td>18247</td>
<td></td>
</tr>
<tr>
<td>PCT3</td>
<td>6</td>
<td>10.0</td>
<td>15.0</td>
<td>0.996</td>
<td>240</td>
<td>240</td>
<td>506.5</td>
<td>121560</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Results for the 2-mode prostate cancer treatment model ($k = 2$). For each sample generated, $SReach$ analyzed systems with 41 variables and 10 ODEs in the unfolded SMT formulae.
**Synthesized KillerRed 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. One approach to antibiotic resistance is to engineer a temperate phage λ with light-activated production of superoxide (SOX). The incorporated KillerRed protein is phototoxic and provides another level of controlled bacteria killing [21]. A PHA with subtle non-determinism for this synthesized KillerRed model (as shown in Figure 2) has been constructed. Considering individual differences of bacterial cells and distinct experimental environments, additional randomness on transition probabilities have been considered. SReach was used to validate this model by estimating the probabilities of killing bacterial cells with different ks (see Table 3). We noticed that the probabilities of paths going through mode 6 to mode 11 are close to 0. This remains even after increasing the probability of entering mode 6, indicating that it is impossible for this model to enter mode 6. SReach was also used to find out (a) 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 with probability larger than 0.5 (see the first two rows of Table 4), (b) that the lower bound for the duration of exposure to light is 3 for successful bacterial killing with with probability larger than 0.5 (see row 3-4 of Table 4), (c) that the time to remove IPTG is insensitive considering whether bacterial cells will be killed with probability larger than 0.5 (see row 5-6 of Table 4), and (d) that the upper bound of the necessary concentration of SOX to kill bacterial cells, with probability larger than 0.5, is 0.6667 (see from row...
All these findings have been reported to biologists for further checking.

<table>
<thead>
<tr>
<th>k</th>
<th>Est_P</th>
<th>#S_S</th>
<th>#T_S</th>
<th>A_T(s)</th>
<th>T_T(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>0.544</td>
<td>8951</td>
<td>16452</td>
<td>0.074</td>
<td>219.38</td>
</tr>
<tr>
<td>6</td>
<td>0.247</td>
<td>3045</td>
<td>12336</td>
<td>0.969</td>
<td>11957</td>
</tr>
<tr>
<td>7</td>
<td>0.096</td>
<td>559</td>
<td>5808</td>
<td>5.470</td>
<td>31770</td>
</tr>
</tbody>
</table>

Table 3: Results for the 11-mode killerred model.

<table>
<thead>
<tr>
<th>t_{lightON} (t.u.)</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_{total} (t.u.)</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>

| t_{lightOFF} (t.u.) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| killed bacteria cells | failed | failed | failed | succ | succ | succ | succ | succ | succ | succ |

| t_{rmIPTG} (t.u.) | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| killed bacteria cells | succ | succ | succ | succ | succ | succ | succ | succ | succ | succ |

| SOXthres (M) | 1e-4 | 2e-4 | 3e-4 | 4e-4 | 5e-4 | 6e-4 | 7e-4 | 8e-4 | 9e-4 | 1e-3 |
| t_{total} (t.u.) | 5.1 | 5.2 | 5.4 | 17 | 19 | 48 | 61 | 71 | 36 | 42 |

Table 4: Formal analysis results for our KillerRed hybrid model.

Additional benchmarks. To further demonstrate SReach’s applicability, we also applied it to additional benchmarks including HA_p, PHAs, and PHA_r with subtle non-determinism. Table 5 shows the results of these experiments. 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 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 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 PHA_r with trivial non-determinism - KR (our killerred models). Moreover, the detailed description of some of additional benchmarks and above case studies are presented in Appendix D. The full descriptions of all the models that mentioned in this paper can be found on the tool website.

5 Conclusions and future work

We have presented a tool that combines \( \delta \)-decision procedures and statistical tests. It supports probabilistic bounded \( \delta \)-reachability analysis for HA_p and PHA_r with no or subtle non-determinism. This tool has been used to analyze three representative examples - a prostate cancer treatment model, a cardiac model, and a synthesized KillerRed model - and other benchmarks, 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.
## Table 5: #Ms = number of modes, K indicates the unfolding steps, #ODEs = number of ODEs in the unfolded formulae, #Vs = number of total variables in the unfolded formulae, #RVs = number of random variables in the model, δ = precision used in dReach.

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>#Ms</th>
<th>K</th>
<th>#ODEs</th>
<th>#Vs</th>
<th>#RVs</th>
<th>δ</th>
<th>Est. P</th>
<th>#S</th>
<th>S</th>
<th>A</th>
<th>T(s)</th>
<th>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>572</td>
<td>7126</td>
<td>0.086</td>
<td>612.836</td>
<td></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>
<td></td>
</tr>
<tr>
<td>BBwDv1</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>20</td>
<td>4</td>
<td>0.001</td>
<td>0.208</td>
<td>2206</td>
<td>10919</td>
<td>0.080</td>
<td>873.522</td>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
<tr>
<td>TldK3</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>12.839</td>
<td>2914.448</td>
<td></td>
</tr>
<tr>
<td>DTldK3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>26</td>
<td>2</td>
<td>0.001</td>
<td>0.996</td>
<td>227</td>
<td>227</td>
<td>0.382</td>
<td>86.714</td>
<td></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>0.161</td>
<td>1442</td>
<td>8961</td>
<td>0.253</td>
<td>917.884</td>
<td></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>0.381</td>
<td>5953</td>
<td>15639</td>
<td>0.080</td>
<td>612.836</td>
<td></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>
<td></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>0.004</td>
<td>0</td>
<td>227</td>
<td>0.120</td>
<td>27.240</td>
<td></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>0.215</td>
<td>48.806</td>
<td></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>0.215</td>
<td>48.806</td>
<td></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>9353.058</td>
<td></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>0.228</td>
<td>2662</td>
<td>11077</td>
<td>0.095</td>
<td>1109.315</td>
<td></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>
<td></td>
</tr>
<tr>
<td>QuadC</td>
<td>1</td>
<td>0</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>825.641</td>
<td>187420.507</td>
<td></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>0.524</td>
<td>345</td>
<td>658</td>
<td>5.01</td>
<td>3295.82</td>
<td></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>2.35</td>
<td></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>
<td></td>
</tr>
<tr>
<td>KRk6</td>
<td>6</td>
<td>8</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>
<td></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>0.096</td>
<td>558</td>
<td>5795</td>
<td>16.275</td>
<td>94311.18</td>
<td></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>0</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>0</td>
<td>227</td>
<td>0.015</td>
<td>3.43</td>
<td></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>0</td>
<td>227</td>
<td>0.026</td>
<td>5.92</td>
<td></td>
</tr>
</tbody>
</table>

## References

A  δ-Decisions for Hybrid Models

In order to overcome the undecidability of reasoning about hybrid systems, Gao et al. recently defined the concept of δ-satisfiability over the reals, and presented a corresponding δ-complete decision procedure [13,14]. The main idea is to decide correctly whether slightly relaxed sentences over the reals are satisfiable or not. The following definitions are from [14].

**Definition 5.** A bounded quantifier is one of the following:

\[ \exists^{[a,b]} x = \exists x : (a \leq x \land x \leq b) \]
\[ \forall^{[a,b]} x = \forall x : (a \leq x \land x \leq b) \]

**Definition 6.** A bounded \( \Sigma_1 \) sentence is an expression of the form:

\[ \exists^{I_1} x_1, ..., \exists^{I_n} x_n : \psi(x_1, ..., x_n) \]

where \( I_i = [a_i, b_i] \) are intervals, \( \psi(x_1, ..., x_n) \) is a Boolean combination of atomic formulas of the form \( g(x_1, ..., x_n) \) \( \text{op} \) 0, where \( g \) is a composition of Type 2-computable functions and \( \text{op} \in \{<, \leq, >, \geq, =, \neq\} \).

Note that any bounded \( \Sigma_1 \) sentence is equivalent to a \( \Sigma_1 \) sentence in which all the atoms are of the form \( f(x_1, ..., x_n) = 0 \) (i.e., the only \( \text{op} \) needed is ‘=’). Essentially, Type 2-computable functions can be approximated arbitrarily well by finite computations of a special kind of Turing machines (Type 2 machines); most ‘useful’ functions over the reals are Type 2-computable. The notion of δ-weakening of a bounded sentence is central to δ-satisfiability.

**Definition 7.** Let \( \delta \in \mathbb{Q}^+ \cup \{0\} \) be a constant and \( \phi \) a bounded \( \Sigma_1 \)-sentence in the standard form

\[ \phi = \exists^{I_1} x_1, ..., \exists^{I_n} x_n : \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} f_{ij}(x_1, ..., x_n) = 0 \]  \hspace{1cm} (1)

where \( f_{ij}(x_1, ..., x_n) = 0 \) are atomic formulas. The \( \delta \)-weakening of \( \phi \) is the formula:

\[ \phi^\delta = \exists^{I_1} x_1, ..., \exists^{I_n} x_n : \bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} |f_{ij}(x_1, ..., x_n)| \leq \delta \]  \hspace{1cm} (2)

Note that \( \phi \) implies \( \phi^\delta \), while the converse is obviously not true. The bounded δ-satisfiability problem asks for the following: given a sentence of the form (1) and \( \delta \in \mathbb{Q}^+ \), correctly decide whether

- **unsat:** \( \phi \) is false,
- **δ-sat:** \( \phi^\delta \) is true.
If the two cases overlap either decision can be returned: such a scenario reveals that the formula is fragile — a small perturbation (i.e., a small $\delta$) can change the formula’s truth value.

A qualitative property of hybrid systems that can be checked is bounded $\delta$-reachability. It asks whether the system reaches the unsafe region after $k \in \mathbb{N}$ discrete transitions.

**Definition 8.** Bounded $k$ step $\delta$-reachability in hybrid systems can be encoded as a bounded $\Sigma_1$-sentence

$$
\exists x_{0,q_0}^0, \exists x_{0,q_0}^1, ..., \exists x_{0,q_m}^0, \exists x_{0,q_m}^1, ..., \exists x_{k,q_m}^0, \exists x_{k,q_m}^1, \exists x_{t_0,q_0}^0, \exists x_{t_0,q_0}^1, ..., \exists x_{t_0,q_m}^0, \exists x_{t_0,q_m}^1, \exists x_{t_k,q_m}^0, \exists x_{t_k,q_m}^1,
$$

where

$$
\forall q \in Q
\left(\bigwedge_{i=0}^{k-1} \left( \bigvee_{q, q' \in Q} \left( \text{jump}_{q \to q'}(x_i^q, x_{i+1, q'}) \land \text{flow}_{q'}(x_{i+1, q'}, x_{t_i+1, q'}) \land \text{unsafe}_q(x_{t_i, q}) \right) \right)
\right)
$$

and $q'$ is a successor mode.

Intuitively, the formula above can be understood as follows: the first conjunction is asking for a set of continuous variables which satisfy the initial condition in one of the modes and the flow in that mode; the second conjunction is looking for a set of vectors which satisfy any $k$ discrete jumps and flows in each successor mode defined by the jumps; the third conjunction is verifying whether the state of the system (the mode and the set of continuous variables in the mode after $k$ jumps) belongs to the unsafe region. Note that the previous definition asks for reachability in exactly $k$ steps. One can build a disjunction of formula (3) for all values from 1 to $k$, thereby obtaining reachability within $k$ steps.

The $\delta$-reachability problem can be solved using the described $\delta$-complete decision procedure, which will correctly return one of the following answers:

- **unsat:** the system never reaches the bad region $U$,
- **$\delta$-sat:** the $\delta$-perturbation of (3) is true, and a witness, i.e., an assignment for all the variables, is returned.

## B 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** [20]. As a simple class of sequential tests, it tests the one-sided composite hypotheses $H_0 : \theta \leq \theta_0$ versus $H_1 : \theta \geq \theta_1$ for the natural parameter $\theta$ of an exponential family of distributions under the $0-1$ loss and cost $c$ per
observation. [20] shows that these tests have nearly optimal frequentist properties and also provide approximate Bayes solutions with respect to a large class of priors.

Bayes factor test [19]. 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.

Bayes 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 [28] 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) [27]. 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 $\Lambda_n < A$. It decides that $H_1$ is true and terminates if $\Lambda_n > B$. 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 [17]. 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 [29]. 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. [29] 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 [8], the error $\epsilon$ with a confidence $c$ between the real probability $p$ and the estimated $\hat{p}$ is bounded:

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

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.
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 that 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

\[
\text{<logic_formula1> ==>} @\text{<target_mode> <logic_formula2>},
\]

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

\[
\text{(and <logic_formula1> <stochastic choice>) ==>} @\text{<target_mode> <logic_formula2>},
\]

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 corresponding 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 system parameters) with Bernoulli distribution (B), Uniform distribution (U), Gaussian distribution (N), Exponential distribution (E), and general Discrete distribution with given possible values and corresponding probabilities (DD).

```plaintext
1 # define pi 3.1416
2 N(1,0.1) mu1;
3 U(10,15) thro;
```
Example 2. This example demonstrates the format of the input file for a probabilistic hybrid automaton with additional randomness for transition probabilities. Note that, unlike the notations of declarations of random variables representing system parameters and probabilistic transitions, declarations 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));
```
18 Q. Wang, S. Kong, S. Gao, E. Clarke

(\text{and} (\cos(x) \leq 0) (pjump2 = 1)) \implies @2 (\text{and} (x' = x) (y' = y));

(\text{and} (\cos(x) \leq 0) (pjump2 = 2)) \implies @1 (\text{and} (x' = x) (y' = y));

}\}

{\text{mode 2;}}

\text{invt:}

\begin{align*}
(x & \leq 200); \\
(x & \geq -2.2); \\
(y & \leq 85.1); \\
(y & \geq 2);
\end{align*}

\text{flow:}

\begin{align*}
\frac{dx}{dt} & = x; \\
\frac{dy}{dt} & = 3 \times x - y ^ 2;
\end{align*}

\text{jump:}

(\text{and} (x \leq 1000) (x \geq -1000) (y \leq 1000) (y \geq -1000)) \implies @2 (\text{and} (x' = x) (y' = y));

}\}

\text{init:}

\begin{align*}
@1 & (\text{and} (x \geq 0.1) (x \leq 1.4) (y = 1.1)); \\
@2 & (\text{and} (x \geq -10) (y \geq -10));
\end{align*}

\text{goal:}

\begin{align*}
\text{C.2 Command line}
\end{align*}

\text{SReach offers two choices. It can be run sequentially by typing}

\text{sreach\_sq <statistical\_testing\_option> <filename>}

\text{<dReach> <k> <delta>,}

or in parallel by

\text{sreach\_para <statistical\_testing\_option> <filename>}

\text{<dReach> <k> <delta>,}

where:

- \text{statistical\_testing\_option} is a text file containing a sequence of test specifications. We will introduce the usages of statistical testing options in the following part;
- \text{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 subsection;
- \text{dReach} is a tool for bounded reachability analysis of hybrid systems based on dReal;
- \text{k} is the number of steps of the model that the tool will explore; and
- \text{delta} is the precision for the \(\delta\)-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.

BFT with indifference region: BFTI <theta> <T> <alpha> <beta> <delta>, where, besides the parameters used in the above Bayes factor test, delta is given to create the indifference region - [p₀, p₁], where p₀ = theta - delta and p₁ = theta + delta. Now, it tests H₀: p ≥ p₀ against H₁: p ≤ p₁.

Sequential probability ratio test (SPRT): SPRT <theta> <T> <delta>.

Chernoff-Hoeffding bound: CHB <delta₁> <coverage_probability>, where delta₁ is the given precision, and coverage_probability indicates the confidence.

Bayesian Interval Estimation with Beta prior: BEST <delta₁> <coverage_probability> <alpha> <beta>.

Direct/Naive Sampling: NSAM <num_of_samples>.

D Model description

Synthesized Killerred Model. The ODEs missing in Figure 2 are as follows.

\[
\frac{dmRNA}{dt} = k_{RNA_{syn}} \cdot [DNA] - k_{RNA_{deg}} \cdot [mRNA]
\]

\[
\frac{d[KR_{im}]}{dt} = k_{KR_{im_{syn}}} \cdot [mRNA] - (k_{KR_{im}} + k_{KR_{im_{deg}}}) \cdot [KR_{im}]
\]

\[
\frac{d[KR_{mds}]}{dt} = k_{KR_{im}} \cdot [KR_{im}] - k_{KR_{mds_{deg}}} \cdot [KR_{mds}] \text{ (before turning on the light)}
\]

\[
\frac{d[KR_{mdS}]}{dt} = k_{KR_{im}} \cdot [KR_{im}] + k_{KR_{f}} \cdot [KR_{mdS^*}] + k_{KR_{ic}} \cdot [KR_{mdS^*}] + k_{KR_{ic_{deg}}} \cdot [KR_{mdS}]
\]

\[
\frac{d[KR_{mdT^*}]}{dt} = k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_{nrd}} \cdot [KR_{mdT^*}] - k_{KR_{SOX2}} \cdot [KR_{mdT^*}] - k_{KR_{SOX1}} \cdot [KR_{mdT^*}] - k_{KR_{mdT_{deg}}} \cdot [KR_{mdT^*}] - k_{KR_{ex}} \cdot [KR_{mdS}]
\]

(assuming the light is turned on)
\[
\frac{d[SOX]}{dt} = k_{KR_{SOX}d_1} \cdot [KR_{mdT^\ast}] + k_{KR_{SOX}d_2} \cdot [KR_{mdT^\ast}] - \frac{d[SOX_{sod}]}{dt} = k_{SOD} \cdot V_{maxSOD} \cdot [SOX] \\
K_m + [SOX]
\]

**Atrial Fibrillation.** The model has four discrete control locations, four state variables, and nonlinear ODEs. A typical set of ODEs in the model is as follows. The exponential term on the right-hand side of the ODE is the sigmoid function, which often appears in modeling biological switches.

\[
\begin{align*}
\frac{du}{dt} &= e + (u - \theta_v)(s_u - u)vg_{f_1} + wsg_{s_1} - g_{so}(u) \\
\frac{ds}{dt} &= \frac{g_{s_2}}{(1 + \exp(-2k(u - us)))} - g_{s_2}s \\
\frac{dv}{dt} &= -g_{v_1}^+ \cdot v \\
\frac{dw}{dt} &= -g_{w_1}^+ \cdot w
\end{align*}
\]

**Prostate Cancer Treatment.** The nonlinear ODEs in the Prostate-Cancer-Treatment model are as follows.

\[
\begin{align*}
\frac{dx}{dt} &= (\alpha_x(k_1 + (1 - k_1) \frac{z}{z + k_2} - \beta_x((1 - k_3) \frac{z}{z + k_4} + k_3)) - m_1(1 - \frac{z}{z_0}))x + c_1x \\
\frac{dy}{dt} &= m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d \frac{z}{z_0} - \beta_y)y + c_2y \\
\frac{dz}{dt} &= -\frac{z}{\tau} + c_3z \\
\frac{dv}{dt} &= (\alpha_x(k_1 + (1 - k_1) \frac{z}{z + k_2} - \beta_x(k_3 + (1 - k_3) \frac{z}{z + k_4})) - m_1(1 - \frac{z}{z_0}))x + c_1x \\
&\quad + m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d \frac{z}{z_0} - \beta_y)y + c_2y
\end{align*}
\]

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

\[
\begin{align*}
\frac{dx}{dt} &= -ax \cdot \sin(\omega_1 \cdot \tau) \\
\frac{dy}{dt} &= -ay \cdot \sin((\omega_1 + c_1) \cdot \tau) \cdot \sin(\omega_2) \cdot 2 \\
\frac{dz}{dt} &= -az \cdot \sin((\omega_2 + c_2) \cdot \tau) \cdot \cos(\omega_1) \cdot 2 \\
\frac{d\omega_1}{dt} &= -c_3 \cdot \omega_1 \\
\frac{d\omega_2}{dt} &= -c_4 \cdot \omega_2 \\
\frac{d\tau}{dt} &= 1
\end{align*}
\]

**Quadcopter Control.** We developed a model that contains the full dynamics of a quadcopter. We use the model to solve control problems by answering reach-
ability questions. A typical set of the differential equations are the following.

\[
\frac{d\omega_x}{dt} = L \cdot k \cdot (\omega_1^2 - \omega_2^2)(1/I_{xx}) - (I_{yy} - I_{zz})\omega_y\omega_z/I_{xx}
\]

\[
\frac{d\omega_y}{dt} = L \cdot k \cdot (\omega_2^2 - \omega_1^2)(1/I_{yy}) - (I_{zz} - I_{xx})\omega_x\omega_z/I_{yy}
\]

\[
\frac{d\omega_z}{dt} = b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)(1/I_{zz}) - (I_{xx} - I_{yy})\omega_x\omega_y/I_{zz}
\]

\[
\frac{d\phi}{dt} = \omega_x + \frac{\sin(\phi) \sin(\theta)}{\cos(\phi)} \omega_y
\]

\[
+ \frac{\sin(\phi)^2 \cos(\theta)}{\cos(\phi)} + \cos(\phi) \cos(\theta) \omega_z
\]

\[
\frac{d\theta}{dt} = -\frac{\sin(\phi)^2 \cos(\theta)}{\cos(\phi)} \omega_y + \cos(\phi) \cos(\theta) \cos(\phi)^2
\]

\[
+ \frac{1}{\cos(\phi)} \omega_y - \frac{\sin(\phi) \cos(\theta)}{\cos(\phi)} + \cos(\phi) \cos(\theta) \cos(\phi) \omega_z
\]

\[
\frac{d\psi}{dt} = \frac{\sin(\phi)}{\cos(\phi)} \omega_y
\]

\[
+ \frac{\sin(\phi)^2 \cos(\theta)}{\cos(\phi)} + \cos(\phi) \cos(\theta) \omega_z
\]

\[
\frac{dx_p}{dt} = (1/m)(\sin(\theta) \sin(\psi)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot xp)
\]

\[
\frac{dy_p}{dt} = (1/m)(-\cos(\psi) \sin(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot yp)
\]

\[
\frac{dz_p}{dt} = (1/m)(-g - \cos(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot zp)
\]

\[
\frac{dx}{dt} = xp, \frac{dy}{dt} = yp, \frac{dz}{dt} = zp
\]