SReach: A Probabilistic Bounded δ-Reachability Analyzer for Stochastic Hybrid Systems *

Qinsi Wang[1], Paolo Zuliani[2], Soonho Kong[1], 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 δ-complete decision procedures and statistical tests to solve δ-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_ps 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 *SReach* which supports probabilistic bounded δ-reachability analysis for the above two model classes. It combines the recently proposed δ-complete bounded reachability analysis technique [12] with statistical testing techniques. *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 δ-complete analysis method, the full non-determinism of models will be considered. The coverage of simulation will be increased, as the δ-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, *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 *SReach* can answer questions including model validation/falsification, parameter synthesis, and sensitivity analysis. To further demonstrate its applicability, we also apply it to additional real-world hybrid systems with parametric uncertainty.

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 procedure [11]. But their tool SiSAT supports only discrete random variables. Ellen et al. [9] proposed a statistical model checking technique for verifying hybrid systems with continuous non-determinism, thereby expanding the class of systems analyzable, yet confined dynamics to (non-linear) pre-post conditions rather than ODEs. *SReach* supports both discrete and continuous random variables, and ODEs. ProbReach [22] also uses the δ-complete procedures and offers verified estimated probability interval containing the real probability, yet can only deal with hybrid systems with initial random variables. While *SReach* is able to handle probabilistic transitions as well.

The paper proceeds by introducing two model classes of SHSs under consideration in Section 2. Section 3 formally states probabilistic bounded δ-reachability problems and explains how *SReach* solves these problems by combining δ-complete decision procedures with statistical tests. Case studies and additional experiments are discussed in Section 4. Section 5 concludes the paper.

2 Stochastic Hybrid Models

Before introducing the algorithm implemented by *SReach* and the problems that it can handle, we first define two model classes that *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 ($\mathbf{HA}_p$). *A hybrid automaton with parametric uncertainty is a tuple* $H_p = \langle (Q, E),\ V,\ RV,\ \mathsf{Init},\ \mathsf{Flow}, \mathsf{Inv},\ \mathsf{Jump},\ \Sigma \rangle$*, 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* $Init(q)$ *and invariant condition* $Inv(q)$ *are predicates whose free variables are from* $V \cup RV$*, and the flow condition* $Flow(q)$ *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* $V \cup V' \cup RV$*.*
- Σ *is a finite set of events, and an edge labeling function event* $: E \to \Sigma$ *assigns to each control switch an event.*

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

Definition 2 ($\mathbf{PHA}_r$). *A probabilistic hybrid automaton with additional randomness* H_r *consists of* Q, E, V, RV, Init, $\mathsf{Flow}, \mathsf{Inv}$, Σ *as in Definition 1, and Cmds, which is a finite set of probabilistic guarded commands of the form:*
$g \ \to\ p_1 : u_1 \ +\ \cdots\ +\ p_m : u_m$*,*
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 $x \geq 5 \ \to\ p_1 : (x' = sin(x)) + (1 - p_1) : (x' = p_x)$, 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 δ-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 over-approximation can be tight (tunable by an arbitrarily small rational parameter δ), and a false alarm with a small δ 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 δ-reachability problem based on the bounded δ-reachability problem defined in [12] .

Definition 3. *The probabilistic bounded k step δ-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 $\mathbf{r}$, $Pr(\mathbf{r})$ a probability measure over $\mathbf{r}$, and Ω the sample space of $\mathbf{r}$, the reachability probability is $\int_\Omega I_T(\mathbf{r})dPr(\mathbf{r})$, where $I_T(\mathbf{r})$ is the indicator function which is 1 if H_p with $\mathbf{r}$ reaches T in k steps.*

Definition 4. *For a PHA_r H_r, the probabilistic bounded k step δ-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 σ is an execution in the set E.*

Algorithm 1 SReach

```
1: function SReach(MP, ST, δ, k)
2:     if MP is a HA_p then
3:         MP ← EncRM_1(MP)                          ▷ encode uncertain system parameters
4:     else                                                    ▷ otherwise a PHA_r
5:         MP ← EncRM_2(MP)  ▷ encode probabilistic jumps and extra randomness
6:     end if
7:     Succ, N ← 0                          ▷ number of δ-sat samples and total samples
8:     Assgn ← ∅          ▷ record unique sampling assignments and dReach results
9:     RV ← ExtractRV(MP)                  ▷ get the RVs from the probabilistic model
10:    repeat in parallel
11:        S_i ← Sim(RV)                                        ▷ sample the parameters
12:        if S_i ∈ Assgn.sample then
13:            Res ← Assgn(S_i).res                          ▷ no need to call dReach
14:        else
15:            M_i ← Gen(MP, S_i)                          ▷ generate a dReach model
16:            Res ← dReach(M_i, δ, k)      ▷ call dReach to solve k-step δ-reachability
17:        end if
18:        if Res = δ-sat then Succ ← Succ + 1
19:        end if
20:        N ← N + 1
21:    until ST.done(Succ, N)                              ▷ 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 δ-complete analyzer *dReach* is utilized to analyze each intermediate HA M_i, together with the desired precision δ and unfolding depth k. The analyzer returns either unsat or δ-sat for M_i. This information is then used by a chosen statistical testing procedure to decide whether to stop or to repeat the procedure, 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 δ-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 transitions in advance results in an over-approximation of the original PHA$_r$, where safety properties are preserved. To promise a tight over-approximation and correctness 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$_r$s, *SReach* is supposed to be used on models with no or few non-deterministic transitions, 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 reduces 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).

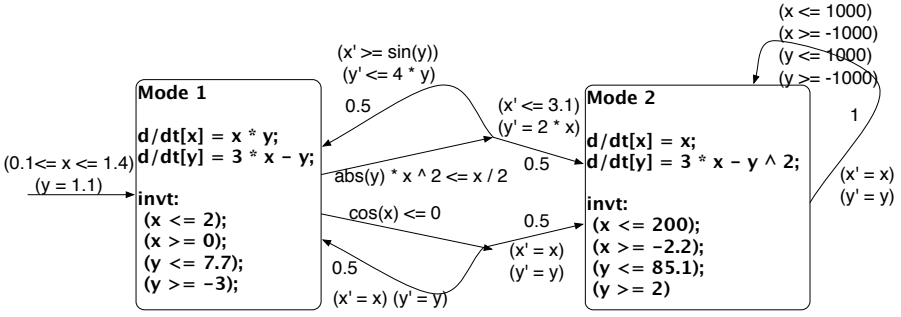

Fig. 1: An example probabilistic hybrid automaton

Currently, *SReach* supports a number of hypothesis testing and statistical estimation 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

[29], 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 δ-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-

ters. 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.

Model	#RVs	EPI_TO1	EPI_TO2	#S_S	#T_S	Est_P	A_T(s)	T_T(s)
Cd_to1_s	1	U(6.1e-3, 7e-3)	6	240	240	0.996	0.270	64.80
Cd_to1_uns	1	U(5.5e-3, 5.9e-3)	6	0	240	0.004	0.042	10.08
Cd_to2_s	1	400	U(0.131, 6)	240	240	0.996	0.231	55.36
Cd_to2_uns	1	400	U(0.1, 0.129)	0	240	0.004	0.038	9.15
Cd_to12_s	2	N(400, 1e-4)	N(6, 1e-4)	240	240	0.996	0.091	21.87
Cd_to12_uns	2	N(5.5e-3, 10e-6)	N(0.11, 10e-5)	0	240	0.004	0.037	8.90

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_S = number of δ-sat samples, #T_S = 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 α_x (the proliferation rate of androgen-dependent (AD) cells), α_y (the proliferation rate of androgen-independent (AI) cells), β_x (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_x to be $U(0.0193, 0.0214)$, α_y to be $U(0.0230, 0.0254)$, β_x to be $U(0.0072, 0.0079)$, β_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.

Model	#RVs	r_0	r_1	Est_P	#S_S	#T_S	A_T(s)	T_T(s)
PCT1	6	5.0	10.0	0.496	8226	16584	0.596	9892
PCT2	6	7.0	11.0	0.994	335	336	54.307	18247
PCT3	6	10.0	15.0	0.996	240	240	506.5	121560

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.

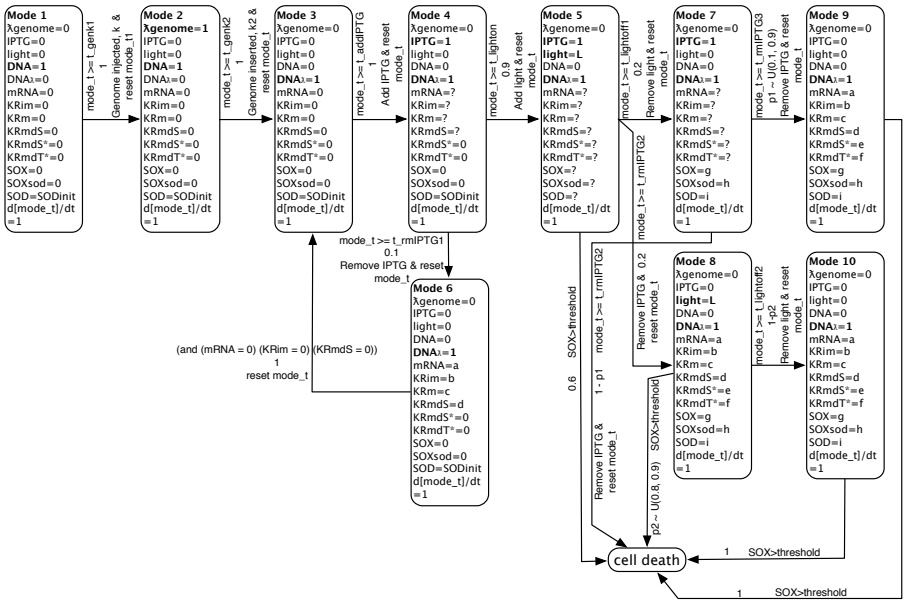

Fig. 2: A probabilistic hybrid automaton for synthesized phage-based therapy model

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_r 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

7-8 of Table 4). All these findings have been reported to biologists for further checking.

k	Est_P	#S_S	#T_S	A_T(s)	T_T(s)	k	Est_P	#S_S	#T_S	A_T(s)	T_T(s)
5	0.544	8951	16452	0.074	1219.38	8	0.004	0	240	0.004	0.88
6	0.247	3045	12336	0.969	11957.12	9	0.004	0	240	0.012	2.97
7	0.096	559	5808	5.470	31770.36	10	0.004	0	240	0.013	3.18

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

$t_{lightON}$ (t.u.)	1	2	3	4	5	6	7	8	9	10
t_{total} (t.u.)	16	17.2	18.5	20	21.3	22.7	23.5	24.1	25	30
$t_{lightOFF_1}$ (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_3} (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
SOX_{thres} (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$_p$s, PHAs, and PHA$_r$s 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 δ-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$_r$s 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 δ-decision procedures and statistical tests. It supports probabilistic bounded δ-reachability analysis for HA$_p$s and PHA$_r$s 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.

Benchmark	#Ms	K	#ODEs	#Vs	#RVs	δ	Est_P	#S_S	#T_S	A_T(s)	T_T(s)
BBK1	1	1	2	14	3	0.001	0.754	5372	7126	0.086	612.836
BBK5	1	5	2	38	3	0.001	0.059	209	3628	0.253	917.884
BBwDv1	2	2	4	20	4	0.001	0.208	2206	10919	0.080	873.522
BBwDv2K2	2	2	4	20	3	0.001	0.845	7330	8669	0.209	1811.821
BBwDv2K8	2	8	4	56	3	0.001	0.207	2259	10901	0.858	9353.058
Tld	2	7	2	33	4	0.001	0.996	227	227	0.213	48.351
Ted	2	7	4	50	4	0.001	0.996	227	227	12.839	2914.448
DTldK3	2	3	4	26	2	0.001	0.996	227	227	0.382	86.714
DTldK5	2	5	4	38	2	0.001	0.161	1442	8961	0.280	2509.078
W4mv1	4	3	8	26	6	0.001	0.381	5953	15639	0.238	3722.082
W4mv2K3	4	3	8	26	6	0.001	0.996	227	227	0.673	152.771
W4mv2K7	4	7	8	50	6	0.001	0.004	0	227	0.120	27.240
DWK1	2	1	4	14	5	0.001	0.996	227	227	0.171	38.817
DWK3	2	3	4	26	5	0.001	0.996	227	227	0.215	48.806
DWK9	2	9	4	62	5	0.001	0.996	227	227	5.144	1167.688
Que	3	2	3	13	4	0.001	0.228	2662	11677	0.095	1109.315
3dOsc	3	2	18	48	2	0.001	0.996	227	227	8.273	1877.969
QuadC	1	0	14	44	6	0.001	0.996	227	227	825.641	187420.507
exPHA01	2	2	4	20	2	0.001	0.524	345	658	5.01	3295.82
exPHA02	2	3	2	17	1	0.001	0.900	5361	5953	0.0004	2.35
KRk5	6	5	84	194	2	0.001	0.544	8946	16457	0.122	2015.64
KRk6	8	6	112	224	6	0.001	0.246	2032	8263	1.385	11444.22
KRk7	10	7	150	271	6	0.001	0.096	558	5795	16.275	94311.18
KRk8	7	8	105	303	6	0.001	0.004	0	227	0.003	0.58
KRk9	9	9	135	335	6	0.001	0.004	0	227	0.015	3.43
KRk10	11	10	165	367	6	0.001	0.004	0	227	0.026	5.92

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*.

References

1. L. Arnold. *Stochastic Differential Equations: Theory and Applications*. Wiley - Interscience, 1974.
2. H. A. Blom, J. Lygeros, M. Everdij, S. Loizou, and K. Kyriakopoulos. *Stochastic hybrid systems: theory and safety critical applications*. Springer, 2006.
3. N. Bruchovsky, L. Klotz, J. Crook, and L. Goldenberg. Locally advanced prostate cancer: biochemical results from a prospective phase ii study of intermittent androgen suppression for men with evidence of prostate-specific antigen recurrence after radiotherapy. *Cancer*, 109(5):858–867, 2007.
4. N. Bruchovsky, L. Klotz, et al. Final results of the Canadian prospective phase ii trial of intermittent androgen suppression for men in biochemical recurrence after radiotherapy for locally advanced prostate cancer. *Cancer*, 107(2):389–395, 2006.
5. A. Bueno-Orovio, E. M. Cherry, and F. H. Fenton. Minimal model for human ventricular action potentials in tissue. *J. of Theor. Biology*, 253(3):544–560, 2008.
6. E. M. Clarke and P. Zuliani. Statistical model checking for cyber-physical systems. In *ATVA*, pages 1–12. Springer, 2011.

7. L. Cordeiro, B. Fischer, and J. Marques-Silva. Smt-based bounded model checking for embedded ansi-c software. *Software Engineering, IEEE*, 38(4):957–974, 2012.
8. R. Durrett. *Probability: theory and examples*. Cambridge University Press, 2010.
9. C. Ellen, S. Gerwinn, and M. Fränzle. Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. *International Journal on Software Tools for Technology Transfer*, pages 1–20.
10. M. Fränzle, E. M. Hahn, H. Hermanns, N. Wolovick, and L. Zhang. Measurability and safety verification for stochastic hybrid systems. In *HSCC*, pages 43–52, Apr. 2011.
11. M. Fränzle, H. Hermanns, and T. Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In *HSCC*, pages 172–186. Springer, 2008.
12. S. Gao, S. Kong, W. Chen, and E. M. Clarke. δ-complete analysis for bounded reachability of hybrid systems. *CoRR*, arXiv:1404.7171, 2014.
13. S. Gao, S. Kong, and E. M. Clarke. dReal: An SMT solver for nonlinear theories over the reals. In *CADE*, pages 208–214. Springer, 2013.
14. S. Gao, S. Kong, and E. M. Clarke. Satisfiability modulo ODEs. In *FMCAD*, pages 105–112, Oct. 2013.
15. E. M. Hahn, G. Norman, D. Parker, B. Wachter, and L. Zhang. Game-based abstraction and controller synthesis for probabilistic hybrid systems. In *QEST*, pages 69–78. IEEE, 2011.
16. T. A. Henzinger. *The theory of hybrid automata*. Springer, 2000.
17. W. Hoeffding. Probability inequalities for sums of bounded random variables. *J American Statistical Association*, 58(301):13–30, 1963.
18. J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *HSCC*, pages 160–173. Springer, 2000.
19. R. E. Kass and A. E. Raftery. Bayes factors. *JASA*, 90(430):773–795, 1995.
20. T. L. Lai. Nearly optimal sequential tests of composite hypotheses. *AOS*, 16(2):856–886, 1988.
21. N. Miskov-Zivanov, Q. Wang, C. Telmer, and E. M. Clarke. Formal analysis provides parameters for guiding hyperoxidation in bacteria using phototoxic proteins. Technical Report CMU-CS-14-137, CMU, 2014.
22. F. Shmarov and P. Zuliani. Probreach: Verified probabilistic delta-reachability for stochastic hybrid systems. In *HSCC*, 2015, to appear.
23. J. Sproston. Decidable model checking of probabilistic hybrid automata. In *FTRTFT*, pages 31–45. Springer, 2000.
24. J. Sproston. Model checking for probabilistic timed and hybrid systems. In *PhD thesis*. SCS, University of Birmingham, 2001.
25. G. Tanaka, Y. Hirata, L. Goldenberg, N. Bruchovsky, and K. Aihara. Mathematical modelling of prostate cancer growth and its application to hormone therapy. *Phil. Trans. Roy. Soc. A: Math., Phys. and Eng. Sci.*, 368(1930):5029–5044, 2010.
26. C. Tinelli. SMT-based model checking. In *NASA FM*, page 1, 2012.
27. A. Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.
28. H. L. Younes. Verification and planning for stochastic processes with asynchronous events. Technical report, DTIC Document, 2005.
29. P. Zuliani, A. Platzer, and E. M. Clarke. Bayesian statistical model checking with application to stateflow/simulink verification. *Formal Methods in System Design*, 43(2):338–367, 2013.

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:*

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

Definition 6. *A bounded Σ_1 sentence is an expression of the form:*

$$\exists^{I_1}x_1, ..., \exists^{I_1}x_n : \psi(x_1, ..., x_n)$$

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

Note that any bounded Σ_1 sentence is equivalent to a Σ_1 sentence in which all the atoms are of the form $f(x_1, ..., x_n) = 0$ (*i.e.*, the only `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 ϕ a bounded Σ_1-sentence in the standard form*

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

where $f_{ij}(x_1, ..., x_n) = 0$ are atomic formulas. The δ-weakening *of ϕ is the formula:*

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

Note that ϕ implies ϕ^δ, 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**: ϕ is false,
- **δ-sat**: ϕ^δ 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 δ) can change the formula's truth value.

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

Definition 8. *Bounded k step δ-reachability in hybrid systems can be encoded as a bounded Σ_1-sentence*

$$\begin{aligned}\exists \boldsymbol{x}^0_{0,q_0}, \exists \boldsymbol{x}^t_{0,q_0}, ..., \exists \boldsymbol{x}^0_{0,q_m}, \exists \boldsymbol{x}^t_{0,q_m}, ..., \exists \boldsymbol{x}^0_{k,q_m}, \exists \boldsymbol{x}^t_{k,q_m} : \\ (\bigvee_{q\in Q} (init_q(\boldsymbol{x}^0_{0,q}) \wedge flow_q(\boldsymbol{x}^0_{0,q}, \boldsymbol{x}^t_{0,q}))) \\ \wedge(\bigwedge_{i=0}^{k-1} (\bigvee_{q,q'\in Q} (jump_{q\to q'}(\boldsymbol{x}^t_{i,q}, \boldsymbol{x}^0_{i+1,q'}) \\ \wedge(flow_{q'}(\boldsymbol{x}^0_{i+1,q'}, \boldsymbol{x}^t_{i+1,q'}))) \wedge (\bigvee_{q\in Q} unsafe_q(\boldsymbol{x}^t_{k,q}))))\end{aligned} \tag{3}$$

where $\boldsymbol{x}^0_{i,q}$ and $\boldsymbol{x}_{i,q}$ represent the continuous state in the mode q at the depth i, 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 δ-reachability problem can be solved using the described δ-complete decision procedure, which will correctly return one of the following answers:

- **unsat**: the system never reaches the bad region U,
- **δ-sat**: the δ-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 θ 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 α (respectively, β). 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 Λ_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 Λ_{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 δ' and coverage probability α, the Chernoff-Hoeffding bound computes a value p' such that $|p' - p| \leq \delta'$ with probability at least α.

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 ϵ with a confidence c between the real probability p and the estimated $\hat{p}$ is bounded:
$\epsilon = \phi^{-1}\left(\frac{c+1}{2}\right)\sqrt{\frac{p(1-p)}{N}}$
where $\phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-x}^{x} e^{-t^2/2}dt$. That is, as N goes to ∞, 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

```
<logic_formula1>  ==>
            @<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

```
(and <logic_formula1> <stochastic choice>)  ==>
            @<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).

```
1  #define pi 3.1416
2  N(1,0.1) mu1;
3  U(10,15) thro;
```

```
E(0.49) theta1;
B(0.75) xinit;
DD(0:0.7, 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(0-((x-mu1+mu2)^2)/(2*theta1^2));
  jump:
        (x>=(thre1+5))==>@2(x'=x);
}
init:
@1      (x=xinit);
goal:
@4      (x>=50);
```

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.

```
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));
```

```
21          (and (cos(x) <= 0) (pjump2 = 1)) ==> @2 (and (x' = x)
                 (y' = y));
22          (and (cos(x) <= 0) (pjump2 = 2)) ==> @1 (and (x' = x)
                 (y' = y));
23  }
24
25  {
26    mode 2;
27    invt:
28          (x <= 200);
29          (x >= -2.2);
30          (y <= 85.1);
31          (y >= 2);
32    flow:
33          d/dt[x] = x;
34          d/dt[y] = 3 * x - y ^ 2;
35    jump:
36          (and (x <= 1000) (x >= -1000) (y <= 1000) (y >=
                -1000)) ==> @2 (and (x' = x) (y' = y));
37  }
38  init:
39  @1      (and (x >= 0.1) (x <= 1.4) (y = 1.1));
40
41  goal:
42  @2      (and (x >= -10) (y >= -10));
```

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 following 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 subsection;
- `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.

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 - $[p_0, p_1]$, where $p_0 =$ `theta - delta` and $p_1 =$ `theta + delta`. Now, it tests $H_0 : p \geq p_0$ against $H_1 : p \leq p_1$.

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/Naïve Sampling: `NSAM <num_of_samples>`.

D Model description

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

$$\frac{\mathrm{d}[mRNA]}{\mathrm{d}t} = k_{RNAsyn} \cdot [DNA] - k_{RNAdeg} \cdot [mRNA]$$

$$\frac{\mathrm{d}[KR_{im}]}{\mathrm{d}t} = k_{KR_{im}syn} \cdot [mRNA] - (k_{KR_m} + k_{KR_{im}deg}) \cdot [KR_{im}]$$

$$\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} = k_{KR_m} \cdot [KR_{im}] - k_{KR_{mdS}deg} \cdot [KR_{mdS}] \ (before\ turning\ on\ the\ light)$$

$$\begin{aligned}\frac{\mathrm{d}[KR_{mdS}]}{\mathrm{d}t} = {} & k_{KR_m} \cdot [KR_{im}] + k_{KR_f} \cdot [KR_{mdS^*}] + k_{KR_{ic}} \cdot [KR_{mdS^*}] + k_{KR_{nrd}} \\ & \cdot [KR_{mdT^*}] + k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] - k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_{mdS}deg} \\ & \cdot [KR_{mdS}] \quad (after\ adding\ light)\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}[KR_{mdS^*}]}{\mathrm{d}t} = {} & k_{KR_{ex}} \cdot [KR_{mdS}] - k_{KR_f} \cdot [KR_{mdS^*}] - k_{KR_{ic}} \cdot [KR_{mdS^*}] \\ & - k_{KR_{isc}} \cdot [KR_{mdS^*}] - k_{KR_{mdS^*}deg} \cdot [KR_{mdS^*}]\end{aligned}$$

$$\begin{aligned}\frac{\mathrm{d}[KR_{mdT^*}]}{\mathrm{d}t} = {} & k_{KR_{isc}} \cdot [KR_{mdS^*}] - k_{KR_{nrd}} \cdot [KR_{mdT^*}] - k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] \\ & - k_{KR_{SOXd2}} \cdot [KR_{mdT^*}] - k_{KR_{mdT^*}deg} \cdot [KR_{mdT^*}]\end{aligned}$$

$$\frac{\mathrm{d}[SOX]}{\mathrm{d}t} = k_{KR_{SOXd1}} \cdot [KR_{mdT^*}] + k_{KR_{SOXd2}} \cdot [KR_{mdT^*}] - \frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t}$$
$$\frac{\mathrm{d}[SOX_{sod}]}{\mathrm{d}t} = k_{SOD} \cdot V_{maxSOD} \cdot \frac{[SOX]}{K_m + [SOX]}$$

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.

$$\frac{du}{dt} = e + (u - \theta_v)(u_u - u)vg_{fi} + wsg_{si} - g_{so}(u)$$
$$\frac{ds}{dt} = \frac{g_{s2}}{(1 + \exp(-2k(u - us)))} - g_{s2}s$$
$$\frac{dv}{dt} = -g_v^+ \cdot v \qquad \frac{dw}{dt} = -g_w^+ \cdot w$$

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

$$\frac{dx}{dt} = (\alpha_x(k_1 + (1 - k_1)\frac{z}{z + k_2} - \beta_x((1 - k_3)\frac{z}{z + k_4} + k_3)) - m_1(1 - \frac{z}{z_0}))x + c_1x$$
$$\frac{dy}{dt} = m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d\frac{z}{z_0}) - \beta_y)y + c_2y$$
$$\frac{dz}{dt} = \frac{-z}{\tau} + c_3z$$
$$\frac{dv}{dt} = (\alpha_x(k_1 + (1 - k_1)\frac{z}{z + k_2} - \beta_x(k_3 + (1 - k_3)\frac{z}{z + k_4})) - m_1(1 - \frac{z}{z_0}))x + c_1x$$
$$+m_1(1 - \frac{z}{z_0})x + (\alpha_y(1 - d\frac{z}{z_0}) - \beta_y)y + c_2y$$

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

$$\frac{dx}{dt} = -ax \cdot sin(\omega_1 \cdot \tau)$$
$$\frac{dy}{dt} = -ay \cdot sin((\omega_1 + c_1) \cdot \tau) \cdot sin(\omega_2) \cdot 2$$
$$\frac{dz}{dt} = -az \cdot sin((\omega_2 + c_2) \cdot \tau) \cdot cos(\omega_1) \cdot 2$$
$$\frac{\omega_1}{dt} = -c_3 \cdot \omega_1 \quad \frac{\omega_2}{dt} = -c_4 \cdot \omega_2 \quad \frac{d\tau}{dt} = 1$$

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.

$$
\begin{aligned}
\frac{\mathrm{d}\omega_x}{\mathrm{d}t} &= L \cdot k \cdot (\omega_1^2 - \omega_3^2)(1/I_{xx}) - (I_{yy} - I_{zz})\omega_y\omega_z/I_{xx} \\
\frac{\mathrm{d}\omega_y}{\mathrm{d}t} &= L \cdot k \cdot (\omega_2^2 - \omega_4^2)(1/I_{yy}) - (I_{zz} - I_{xx})\omega_x\omega_z/I_{yy} \\
\frac{\mathrm{d}\omega_z}{\mathrm{d}t} &= b \cdot (\omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2)(1/I_{zz}) - (I_{xx} - I_{yy})\omega_x\omega_y/I_{zz} \\
\frac{\mathrm{d}\phi}{\mathrm{d}t} &= \omega_x + \frac{\sin(\phi)\sin(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y \\
&\quad + \frac{\sin(\theta)}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z \\
\frac{\mathrm{d}\theta}{\mathrm{d}t} &= -\Big(\frac{\sin(\phi)^2\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)}\omega_y + \cos(\phi)\cos(\theta)\right)\cos(\phi)^2} \\
&\quad + \frac{1}{\cos(\phi)}\Big)\omega_y - \frac{\sin(\phi)\cos(\theta)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_z \\
\frac{\mathrm{d}\psi}{\mathrm{d}t} &= \frac{\sin(\phi)}{\left(\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)\right)\cos(\phi)}\omega_y \\
&\quad + \frac{1}{\frac{\sin(\phi)^2\cos(\theta)}{\cos(\phi)} + \cos(\phi)\cos(\theta)}\omega_z \\
\frac{\mathrm{d}xp}{\mathrm{d}t} &= (1/m)(\sin(\theta)\sin(\psi)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot xp) \\
\frac{\mathrm{d}yp}{\mathrm{d}t} &= (1/m)(-\cos(\psi)\sin(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot yp) \\
\frac{\mathrm{d}zp}{\mathrm{d}t} &= (1/m)(-g - \cos(\theta)k(\omega_1^2 + \omega_2^2 + \omega_3^2 + \omega_4^2) - k \cdot d \cdot zp \\
\frac{\mathrm{d}x}{\mathrm{d}t} &= xp, \frac{\mathrm{d}y}{\mathrm{d}t} = yp, \frac{\mathrm{d}z}{\mathrm{d}t} = zp
\end{aligned}
$$