# The *SReach* Tool

Qinsi Wang

Computer Science Department, Carnegie Mellon University, USA

## 1 Input format

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

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

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

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

```
<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).

```
#define pi 3.1416
N(1,0.1) mu1;
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 addtional 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));
```

```
20          (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));
```

## 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 $\delta$-decision problem.

## 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>`.