The Workshops of the Thirtieth AAAI Conference on Artificial Intelligence
Planning for Hybrid Systems: Technical Report WS-16-12

SMT-Based Reasoning for Uncertain Hybrid Domains

Fedor Shmarov and **Paolo Zuliani**
School of Computing Science, Newcastle University, Newcastle upon Tyne, UK, NE1 7RU
{f.shmarov, paolo.zuliani}@ncl.ac.uk

Abstract

Many practical applications (*e.g.*, plannning for cyber-physical systems) require reasoning about hybrid domains that contain both probabilistic and nondeterministic parametric uncertainty. In general, this is an undecidable problem. We use δ-satisfiability to sidestep undecidability, and we develop an algorithm that computes an enclosure for the range of probability of reaching a goal region in a given number of discrete steps. We utilize SMT techniques that enable reasoning in a safe way, *i.e.*, the computed enclosure is formally guaranteed to contain the reachability probability. We demonstrate the usefulness of our technique on challenging nonlinear hybrid domains.

1 Introduction

Reasoning about hybrid domains is a very important and difficult problem. Hybrid domains integrate continuous evolution with discrete computation, and are much used, *e.g.*, to model cyber-physical systems (anti-lock brakes in cars, biomedical devices, *etc.*). However, it is well known that even basic questions, such as reachability, are undecidable for anything but the simplest hybrid systems (*e.g.*, reachability is PSPACE-complete for timed automata (Alur and Dill 1990)). Also, models of realistic systems typically contain uncertainty due, for example, to faulty components, random environmental effects, and user/adversarial behaviors.

In this paper we consider nonlinear hybrid domains in which the initial conditions are subject to random uncertainty (probability) and/or non-quantified uncertainty (nondeterminism). For such systems we study the probabilistic reachability problem, *i.e.*, computing the probability of reaching a goal region in a given time and number of discrete steps. Note that when probability and nondeterminism are both present, the model will feature not just a single reachability probability, but rather a *range* of reachability probabilities (depending on the nondeterministic initial conditions).

Since for nonlinear hybrid domains even standard reachability is undecidable, we relax the problem by using the notion of δ-satisfiability over the reals (Gao, Avigad, and Clarke 2012). A δ-complete decision procedure can check δ-satisfiability of general first-order real formulae (terms can include continuous functions and solutions of nonlinear differential equations). In particular, δ-satisfiability relaxes standard satisfiability in such a way that when a formula is "too close to call", then any decision is allowed. Essentially, δ-satisfiability sidesteps undecidability by allowing numerical reasoning with errors bounded by an arbitrary positive rational number δ.

In this work we develop an algorithm that exploits δ-complete decision procedures to compute a *numerically guaranteed* enclosure for the range of reachability probabilities of a (possibly nonlinear) hybrid system with probabilistic and nondeterministic parametric uncertainty. We show that in certain parameter scenarios the tightness of the enclosure can be controlled by the user. Furthermore, our algorithm can be used to synthesize formally guaranteed plans. We have validated our approach against Monte Carlo simulation on realistic and challenging hybrid domains featuring nonlinear differential equations.

The structure of the paper is as follows: we start by giving a brief background on the notion of δ-satisfiability, and then we define parametric hybrid automata and the corresponding probabilistic bounded reachability problem. Next we present our algorithm for computing enclosures of the reachability probability ranges for parametric hybrid automata. Finally, we empirically evaluate our technique on two hybrid domains featuring both probabilistic and nondeterministic uncertainty.

2 Parametric Hybrid Automata

In this paper we follow (Gao et al. 2014; Bryce et al. 2015) and encode parametric hybrid models by first-order formulae over the reals (so called $\mathcal{L}_{\mathbb{R}}$-formulae (Gao, Avigad, and Clarke 2012)) whose syntax is as follows

$$t := c \mid x \mid f(t(x));$$
$$\varphi := t(x) > 0 \mid t(x) \geqslant 0 \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \exists x \varphi \mid \forall x \varphi$$

where c is a constant, x a variable, and f a Type 2 computable function. Informally, a real function is Type 2 computable if it can be algorithmically computed with arbitrary precision (most common continuous functions are Type 2 computable — see (Ko 1991) for more details). An $\mathcal{L}_{\mathbb{R}}$-sentence is *bounded* when all its variables are quantified over bounded real intervals.

The notion of weakening is key: given a bounded $\mathcal{L}_{\mathbb{R}}$-sentence φ and $\delta \in \mathbb{Q}^+ \cup \{0\}$, the δ-*weakening* φ^δ of φ is obtained by replacing atoms $t > 0$ and $t \geqslant 0$ by $t > -\delta$ and $t \geqslant -\delta$, respectively. In (Gao, Avigad, and Clarke 2012), the authors presented an algorithm that can correctly decide deciding whether φ^δ is true or φ is false — so-called δ-*satisfiability* of φ. When the two cases overlap, *i.e.*, φ is δ-true and false, then either answer can be returned. Note that answers must be correct: if the returned answer is "φ false", then it must actually be that φ is false; on the other hand, from the answer "φ^δ true" we cannot say whether φ is actually true. Algorithms for δ-satisfiability are called δ-complete decision procedures. The dReal (Gao, Kong, and Clarke 2013) SMT solver implements a δ-complete decision procedure for real formulae over a large class of Type 2 computable real functions, including solutions of nonlinear ordinary differential equations (ODEs).

We now extend the hybrid automaton definition of (Bryce et al. 2015) to our case.

Definition 1. A *parametric hybrid automaton* is a tuple

$$\langle X, \Lambda, Q, \{\mathsf{flow}_q(\lambda, x^0, x^t, t) : q{\in}Q\}, \{\mathsf{inv}_q(\lambda, x) : q{\in}Q\}, \{\mathsf{jump}_{q\to q'}(\lambda, x^t, x^0) : q, q'{\in}Q\}, \mathsf{init}_{q_0}(\lambda, x)\rangle$$

where $X \subset \mathbb{R}^n$ is the *state space*, $\Lambda \subset \mathbb{R}^p$ is the *parameter space*, $Q = \{q_0, \ldots q_m\}$ is the set of modes, and the other components are finite sets of quantifier-free $\mathcal{L}_{\mathbb{R}}$-formulae that define *deterministic* flows, jumps, and an initial state. Both X and Λ are *compact* sets.

A parameter vector $\lambda \in \Lambda$ is thought to be assigned initially and remains fixed throughout the system evolution. Each parameter is assigned either probabilistically or nondeterministically: probabilistic parameters can be defined over unbounded intervals, *e.g.*, normally distributed, while nondeterministic parameters must be defined over bounded intervals. Random parameters defined over unbounded domains are reduced to bounded domains as explained later.

Next, the $\mathcal{L}_{\mathbb{R}}$ encoding of the k-step M-delay reachability for parametric hybrid automata is a simple modification of the one presented in (Gao et al. 2014). As in bounded model checking (Biere et al. 1999), we want a sentence that is true if for some parameter vector the goal is reachable at the k-th transition following the hybrid dynamics

Definition 2. The k-*step reachability* property for a parametric hybrid automaton and an $\mathcal{L}_{\mathbb{R}}$-formula goal_{q_G} is the bounded $\mathcal{L}_{\mathbb{R}}$-sentence $\exists^\Lambda \lambda\, \varphi(\lambda)$, where φ is formula (1).

The notation $\exists^\Lambda \lambda$ is a shorthand for $\exists \lambda \in \Lambda$. The first line states the quantification for each variable in each step; the second line constrains the variables in the initial mode using the flow and the invariant; the third and fourth lines constrains the possible mode transitions at each step and the variables using again the flow and the invariant. The fourth line also encodes the goal that must be reached at step k. (Also, $h_k(Q)$ is the set of all possible mode transitions, and literal enforce is used to make sure that only the correct modes are selected. More details are provided in (Gao et al. 2014).) In our implementation we use the dReach tool (Kong et al. 2015), which generates φ from an hybrid automaton specification and then uses dReal to check its δ-satisfiability. In particular, dReach will correctly return one of two answers about the sentence $\exists^\Lambda \lambda\, \varphi(\lambda)$: unsat, meaning that the sentence is unsatisfiable (the system cannot reach a state in the k-th step satisfying goal_{q_G}); δ-sat, *i.e.*, the sentence is δ-satisfiable.

3 Probabilistic Reachability

We now define the probabilistic version of k-step reachability. Since parameters may also be nondeterministic, this entails that in general the reachability probability will be a function of the nondeterministic parameters. We divide the parameter space Λ of a hybrid automaton into two components: the *random parameters* Λ_R and the *nondeterministic parameters* Λ_N, where $\Lambda = \Lambda_R \times \Lambda_N$. As such, a vector $\lambda \in \Lambda$ may be written as (ρ, ν) for $\rho \in \Lambda_R$ and $\nu \in \Lambda_N$.

Definition 3. Given a parametric hybrid automaton, the k-*step reachability probability* for $\nu \in \Lambda_N$ is the function

$$\mathbf{p}(\nu) = \int_{G(\nu)} dP \tag{2}$$

where

$$G(\nu) = \{\rho \in \Lambda_R : \varphi(\rho, \nu)\}, \tag{3}$$

φ is the k-step reachability property, and P is the probability measure of the random parameters (*i.e.*, $\int_{\Lambda_R} dP = 1$).

Informally, $G(\nu)$ is the set of all random parameters values for which the system (with nondeterministic parameter ν) reaches the goal in k steps. Note that by Definition 1 the dynamics of a parametric hybrid automaton is fully deterministic — it depends only on the parameters Λ and the initial state init_{q_0}.

Note that $\mathbf{p}(\nu)$ needs not to be a continuous function of $\nu \in \Lambda_N$. Because of hybrid dynamics and discrete randomness, $\mathbf{p}$ can have jumps, *i.e.*, points for which the right and left limits differ. On the other hand, if $\mathbf{p}$ does not depend on ν then it is a constant function. In this paper we aim at computing enclosures for the range of function $\mathbf{p}$.

Definition 4. Given $Y \subseteq \Lambda_N$, an *enclosure* for the range of $\mathbf{p}$ is an interval $[a, b]$ such that

$$\forall \nu \in Y \quad \mathbf{p}(\nu) \in [a, b].$$

Note that enclosures depend on the given nondeterministic parameter set Y. It is easy to build examples for which it is necessarily $a = 0$ and $b = 1$, and $\mathbf{p}$ takes any value in [0,1]. By reducing Y it is sometimes possible to reduce the size of the enclosure (*e.g.*, if $\mathbf{p}$ is strictly monotonically increasing in the one-dimensional parameter ν) but this is not possible in general. However, there can be cases for which the enclosure can be made arbitrarily tight, in the sense that a and b are close to the infimum and supremum of the range of $\mathbf{p}$ over Y.

4 Computing Probability Enclosures

In Algorithm 1 we present our technique for computing probabilistic reachability enclosures in parametric hybrid automata. For space reasons we present the algorithm for

$$
\begin{aligned}
\varphi(\lambda) = \quad & \exists^X x_0^0 \exists^X x_0^t \cdots \exists^X x_k^0 \exists^X x_k^t \exists^{[0,M]} t_0 \cdots \exists^{[0,M]} t_k \\
& \mathsf{init}_{q_0}(\lambda, x_0^0) \wedge \mathsf{flow}_{q_0}(\lambda, x_0^0, x_0^t, t_0) \wedge \mathsf{enforce}(q_0, 0) \wedge \forall^{[0,t_0]} t \, \forall^X x \, (\mathsf{flow}_{q_0}(\lambda, x_0^0, x, t) \rightarrow \mathsf{inv}_{q_0}(\lambda, x)) \wedge \\
& \bigwedge_{i=0}^{k-1} \bigvee_{(q,q',i) \in h_k(Q)} \Big(\mathsf{jump}_{q \rightarrow q'}(\lambda, x_i^t, x_{i+1}^0) \wedge \mathsf{flow}_{q'}(\lambda, x_{i+1}^0, x_{i+1}^t, t_{i+1}) \wedge \mathsf{enforce}(q, i) \wedge \mathsf{enforce}(q', i+1) \wedge \\
& \qquad \forall^{[0,t_{i+1}]} t \, \forall^X x \, (\mathsf{flow}_{q'}(\lambda, x_{i+1}^0, x, t) \rightarrow \mathsf{inv}_{q'}(\lambda, x)) \Big) \wedge \mathsf{enforce}(q_G, k) \wedge \mathsf{goal}_{q_G}(x_k^t).
\end{aligned} \tag{1}
$$

the case in which every parameter type (random and nondeterministic) is present in the model. The algorithm takes as input a k-step reachability formula ψ (Definition 2) and a domain of continuous nondeterministic parameters Λ_N, and returns a list of enclosures for the reachability probability (as per Definition 4). Such a list will be indexed by a finite set of disjoint (possibly multi-dimensional) parameter boxes that will cover Λ_N, *i.e.*, to each parameter box there will be associated a probability enclosure. If the model has no nondeterministic parameters, then of course only one enclosure is returned, and its size is bounded above by the ϵ input. In general, if nondeterministic parameters are present the size of the enclosure(s) cannot be controlled by ϵ. This is because the reachability probability function $\mathbf{p}$ (Definition 3) might have jumps, hence the enclosure's size cannot be reduced by refining the nondeterministic parameters box. The algorithm thus has an input ϵ_N that limits the size of the smallest nondeterministic parameter box that will be analyzed.

The algorithm works by going through all (finite) possible combinations of random discrete parameters and then alternate partitioning the continuous random and nondeterministic parameters in order to try to reduce the size of the computed enclosures. A key part of the algorithm is the function eval (line 13), which evaluates reachability and is used to decide how the continuous random and nondeterministic parameter boxes are going to be processed. The function is shown in Algorithm 2: essentially, it utilizes (lines 3 and 5) a δ-complete decision procedure to find out whether the goal is reachable for no, all, or some value of the parameters in box b. In line 1 of Algorithm 2, the precision δ to be passed to the δ-complete decision procedure is computed as a fraction (η) of the smallest dimension of box b. This is an important point: δ must be smaller than the smallest dimension of b in order to decide correctly whether b is inside the unsat or sat parameter region. (We empirically found that $\eta = 0.001$ is a good compromise between avoiding spurious δ-sat answers and computational complexity.) In our implementation we use dReal (Gao, Kong, and Clarke 2013), but any other δ-complete decision procedure would be fine.

Formula ψ^c is displayed in (4). It is essentially the same as ψ, except that in the last line it has an implication that checks for the system *not* reaching the goal.

If eval (line 13) returns ***unsat*** then there is no value in $b_N \times b_R$ for which goal_{q_G} is reachable — this is formally correct. So, we can reduce the upper bound of the probability enclosure by $\underline{\int_{b_R} dP}$ (line 16), where P is the probability measure of the continuous random parameters. (Given an interval $I = [a, b]$ we define $\underline{I} = a$ and $\overline{I} = b$.) If eval returns ***sat*** then for every value in $b_N \times b_R$ it is possible to reach goal_{q_G}. Again, this answer is formally correct. Thus, we can increase the lower bound of the enclosure by $\underline{\int_{b_R} dP}$ (line 18). We remark that these answers are formally correct because they rely on the unsat answer from the δ-complete decision procedure, which can be trusted to be actually true. Finally, if eval returns ***undet*** it means that $b_N \times b_R$ is a *mixed* box, *i.e.*, it contains some parameter values for which reachability is true and some others for which it is false. Therefore, we need to refine the parameter boxes and repeat the process. Note that first we refine the (continuous) random parameter boxes (line 20) and then the nondeterministic parameter boxes (line 25), if we have not reached the lower bound ϵ_N on their size (line 22). The final enclosure for a given nondeterministic parameter box is obtained when either the enclosure has length smaller than ϵ, or the size of the nondeterministic parameter box is smaller than ϵ_N (line 22). Otherwise, each dimension of the nondeterministic parameter box is split into two parts (line 25) and the resulting boxes are further analyzed.

Verified integration. Computing probability enclosures (see Definition 4) essentially amounts to computing integrals with bounded error (see, *e.g.*, (Petras 2007) and (Gonnet 2012) for an overview). For use in lines 16 and 18, we need to compute $\int_{b_R} dP$ with precision ϵ, *i.e.*, we need to compute an interval of size smaller than ϵ that contains the true value of $\int_{b_R} dP$. To do so, we provide to Algorithm 1 a finite partition Π_R of the (compact) random parameters domain such that $1 - \sum_{b \in \Pi_R} \int_b dP < \epsilon$.

Unbounded, multiple random parameters. A random parameter with unbounded domain (*e.g.*, Gaussian) cannot be directly used, since it would define an unbounded $\mathcal{L}_{\mathbb{R}}$-formula. However, for any $0 < \epsilon < 1$ it is possible to find a *bounded* region of the random parameter domain over which the integral with respect to the probability measure is larger than $1 - \epsilon$. For example, in the one-dimensional case we can find an interval $I \subset \mathbb{R}$ such that $\int_I dP > 1 - \epsilon$. This means that we can compute the reachability probability $\mathbf{p}(\nu)$ over I instead of the full space $\mathbb{R}$, and the error will be bounded by ϵ. Therefore, we can use the verified integration procedure mentioned above. Again, the same procedure can be used for the case of multiple independent bounded random parameters, provided each parameter is integrated with a stricter precision ξ such that $\epsilon \geq (1 + \xi)^l - 1$, where ϵ is the required precision for computing probability reachability.

Our main result identifies a subclass of parametric hybrid automata for which it is possible to compute the k-step

$$
\begin{aligned}
\varphi^c(\lambda) = \quad & \exists^X x_0^0 \exists^X x_0^t \cdots \exists^X x_k^0 \exists^X x_k^t \exists^{[0,M]} t_0 \cdots \exists^{[0,M]} t_k \\
& \mathsf{init}_{q_0}(\lambda, x_0^0) \wedge \mathsf{flow}_{q_0}(\lambda, x_0^0, x_0^t, t_0) \wedge \mathsf{enforce}(q_0, 0) \wedge \forall^{[0,t_0]} t \, \forall^X x \, (\mathsf{flow}_{q_0}(\lambda, x_0^0, x, t) \rightarrow \mathsf{inv}_{q_0}(\lambda, x)) \wedge \\
& \bigwedge_{i=0}^{k-1} \bigvee_{(q,q',i) \in h_k(Q)} \Big(\mathsf{jump}_{q \rightarrow q'}(\lambda, x_i^t, x_{i+1}^0) \wedge \mathsf{flow}_{q'}(\lambda, x_{i+1}^0, x_{i+1}^t, t_{i+1}) \wedge \mathsf{enforce}(q, i) \wedge \mathsf{enforce}(q', i+1) \\
& \wedge \forall^{[0,t_{i+1}]} t \, \forall^X x \, (\mathsf{flow}_{q'}(\lambda, x_{i+1}^0, x, t) \rightarrow (\mathsf{inv}_{q'}(\lambda, x) \wedge (\mathsf{enforce}(q_G, k) \rightarrow \neg \mathsf{goal}_{q_G}(x)))) \Big)
\end{aligned} \tag{4}
$$

reachability probability with arbitrary precision. Namely, for parametric hybrid systems featuring only random parameters (without nondeterministic parameters) Algorithm 1 returns a probability enclosure of the size not larger than ϵ.

Remarks. Remember that for an event to have zero probability does not necessarily mean that such event cannot *absolutely* occur. This is because (Lebesgue) integration cannot "see" sets with countable numbers of points — they all integrate to zero. That is, if a system reaches the goal (say a bad state) for no value of the random parameters but one, then we cannot say that the system is absolutely safe.

A property is said to be "robust" when its truth value remains invariant under small numerical perturbations. It has been recently shown that safety verification of robust properties is decidable (Ratschan 2014). Also, bounded and robust $\mathcal{L}_{\mathbb{R}}$-sentences are decidable (Gao, Avigad, and Clarke 2012). Here we do not need any robustness assumption, since in our framework we reason about probabilities.

Our algorithm can also be applied to planning in systems that do not feature any randomness (only nondeterminism). In this case we only need to introduce a "dummy" random parameter (*e.g.*, uniformly distributed) such that it does not affect the behavior of the model, and has the only aim of introducing a probability measure. Then whenever the eval function returns ***sat*** or ***unsat*** the probability enclosure is refined to [1,1] or [0,0] respectively. If eval returns ***undet*** the enclosure remains [0,1].

5 Evaluation

We have implemented our technique in C++ and OpenMP, and have evaluated it on two hybrid models: a car deceleration model and a battery load management model.

Car deceleration model. We consider a car accelerating from 0 to 27.78 m/s (100 km/h) and then stopping. This problem is rather nontrivial as the dynamics of the car is governed by nonlinear differential equations and the model features random (*e.g.*, road condition) and nondeterministic (*e.g.*, driver's reaction time) behaviour. The objective of the experiment is to compute enclosures for the probabilities of reaching a target distance depending on the value of nondeterministic parameters.

The model is a three-mode hybrid system inspired by a model presented in (Bryce et al. 2015). In the first mode the car moves with velocity according to the following ODE:

$$v'(t) = \alpha \exp^{-\alpha t + \beta} - c_d v^2(t)$$

where $\alpha = 0.05776$ and $\beta \sim N(4, 0.1)$ are coefficients modelling the acceleration properties of the car, and $c_d = 3.028 \cdot 10^{-4}\, m^{-1}$ is the drag coefficient. When the target velocity 27.78 m/s is reached, the driver needs t_r seconds to react and push the brakes pedal. The reaction time depends on various factors (*e.g.*, driving experience) so we consider it to be nondeterministic ($t_r \in [0.8, 1.5]$). In this "delay" mode the car is not accelerating, and its velocity is governed by the equation $v'(t) = -c_d v^2(t)$. In the final (braking) mode the car is decelerating according to the equation $v'(t) = \mu a_d - c_d v^2(t)$ where $a_d = -5.25\, m/s^2$ is a constant deceleration and $\mu = 1$ is a coefficient modelling normal road conditions (*e.g.*, friction). Throughout the modes the distance S traveled by the car is governed by $S'(t) = v(t)$.

We consider the following scenarios: 1) the car must stop before reaching 300 metres; 2) the car must stop in the $[300, 310]$ metre range. We used the following settings for the algorithm: $\epsilon = 0.01$ and $\epsilon_N = 0.01$. Our results were validated by computing Monte Carlo confidence intervals (of size 0.01 and coverage probability 0.99) using the Chernoff-Hoeffding bound (Hoeffding 1963). The nondeterministic parameter for this experiment (driver's reaction time) was discretized at 10 points.

The probability range of the car stopping within 300 metres having a reaction time within [0.8, 1.5] seconds is [0.266026, 0.468775], and the probability range of stopping within the range of [300, 310] metres is [0.091,0.1224]. The enclosures of the probability values depending on the value of the nondeterministic parameter are plotted in Figure 1 and Figure 2. The CPU times for computing all of the enclosures were the following: 774 minutes for scenario 1) and 1,457 minutes for scenario 2), both on a 2.8GHz, 4-core system.

Multiple battery load management. In this experiment we use our algorithm to synthesize formally guaranteed plans for a battery load management model (Fox, Long, and Magazzeni 2012). Many autonomous systems (*e.g.*, electric prosthetics) require heavy batteries to service a sufficient level of current and power. Sometimes these requirements can be met by utilizing multiple smaller batteries, however not very efficiently as it is possible to extract more energy from a single, large battery than from multiple batteries of the same total capacity. Also, due to their chemical properties, batteries can *recovery*: when the load is applied, the charge is drawn out faster then it is supplied by the chemical reaction. The available charge can be depleted before the total (available and stored in the chemical reaction) charge is used. Hence, finding an efficient load distribution policy

Algorithm 1: Probabilistic reachability enclosures

Input : ψ: k-step reachability formula
$c \in (0,1)$: tuning constant for unbounded RVs
$\epsilon \in \mathbb{Q}^+$: enclosure precision
D: discrete random parameters domain
Π_R: continuous random parameters partition
Λ_N: continuous nondet. parameters domain
$\epsilon_N \in \mathbb{Q}^+$: precision for nondet. parameter box
Output: L: list of (nondet. parameter box, enclosure)

$B_N = \Lambda_N$; $b_N = B_N$.front()
$L[b_N] = [0,1]$ *// initial enclosure*
repeat
  for $b \in B_N$.keys() **do**
    $P_R[b] = \Pi_R$
    $E[b] = [0,1]$ *// temporary enclosures*
  $d = D$.pop()
  repeat
    repeat
      $b_N = B_N$.pop()
      $B_R = P_R[b_N]$
      repeat
        $b_R = B_R$.pop()
        // evaluate reachability
        res = eval(ψ, d, b_N, b_R)
        if *res* == ***unsat*** **then**
          // goal unreachable from $b_N \times b_R$
          $E[b_N] = [\underline{E[b_N]}, \overline{E[b_N]} - \underline{\int_{b_R} dP}]$
        if *res* == ***sat*** **then**
          // goal reachable $\forall \lambda \in b_N \times b_R$
          $E[b_N] = [\underline{E[b_N]} + \underline{\int_{b_R} dP}, \overline{E[b_N]}]$
        if *res* == ***undet*** **then**
          // branch on random parameters
          for $b \in$ branch(b_R) **do** B_R^m.push(b)
      until B_R.empty()
      // $|b|$ = max dimension of box b
      if $(|E[b_N]| \leq \epsilon) \vee (|b_N| \leq \epsilon_N)$ **then**
        $L[b_N] = L[b_N] + E[b_N] \cdot \text{Prob}(d)$
      else
        // branch on nondeterministic param.
        $B_N^m = \text{branch}(b_N)$
        // populate new grid
        for $b \in B_N^m$ **do**
          $E[b] = E[b_N]$ *// enclosures*
          $L[b] = L[b_N]$
          $P_R[b] = B_R^m$ *// partition*
        // b_N was split, so delete
        L.remove(b_N)
      B_R^m.clear(); E.remove(b_N); P_R.remove(b_N)
    until B_N.empty()
    $B_N = B_N^m$; B_N^m.clear()
  until E.empty()
  // bring forward the refined grid
  $B_N = L$.keys()
until D.empty()
// compensate for unbounded RVs
for $b \in B_N$ **do** $L[b] = L[b] + [0, c \cdot \epsilon]$
return L

Algorithm 2: Function eval(ψ, d, b_R, b_N)

Input : ψ: k-step reachability formula
d, b_R, b_N: parameter boxes
Output: goal never/always/sometimes reachable from $b_R \times b_N$

$\delta =$ "min dimension of box b_R" $\cdot \eta$
$b = b_R \times b_N$
if $\psi(\delta, d, b)$ = unsat **then** *// δ-complete decision proc.*
  return unsat
if $\psi^c(\delta, d, b)$ = unsat **then** *// δ-complete decision proc.*
  return sat
return undet *// ψ and ψ^c both δ-sat*

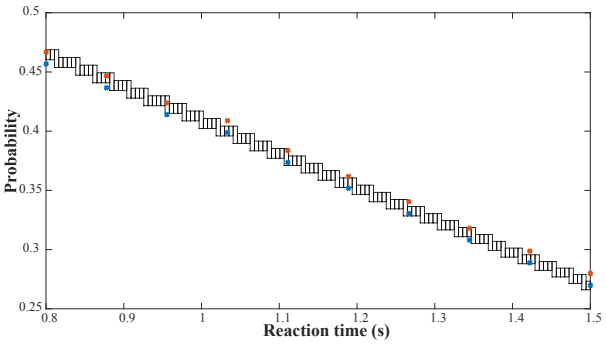

Figure 1: Car deceleration model. Probability enclosure (vs. reaction time) that the car stops within $300m$. The colored dots are confidence intervals bounds.

between the batteries is an important planning problem.

We consider a system model with two batteries introduced in (Fox, Long, and Magazzeni 2012) and modeled as a three-mode hybrid system (see Fig. 3).

Initially, battery one is used for some time t_1 and the second battery remains unused. After that the first battery starts recovering and battery two is used for t_2. Then the second battery is recovering and the first one is used. Then mode 2 and 3 alternate. When the battery is used its dynamics is defined by the system of differential equations:

$$\delta'(t) = \frac{i(t)}{c} \quad \gamma'(t) = -i(t)$$

where γ is the total charge of the battery, δ is the distance between the available-charge well and bound-charge well, $c = 0.166$ is the fraction of the charge in the available-charge well and i is the continuous load. When the battery is recovering its dynamics is governed by the system:

$$\delta'(t) = -\delta(t)k \quad \gamma'(t) = 0$$

where $k = 0.122$ is the conductance of a "valve" between the bound-charge well and the available-charge well. The load applied to the battery in each mode is represented by a differential equation $i'(t) = i(t)$ with initial condition $i(0) = 0.18$ in every mode. The condition for the battery to be "dead" (empty) is $\gamma \leq (1-c)\delta$.

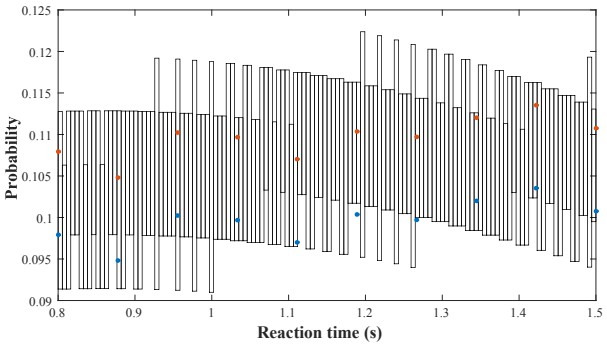

Figure 2: Car deceleration model. Probability enclosure (vs. reaction time) that the car stops inside the range $[300, 310]m$. The colored dots are confidence intervals bounds.

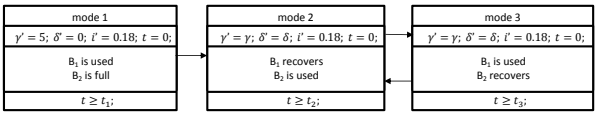

Figure 3: A hybrid model of the two-battery system

We assume that the time for which each battery is used is defined over the interval $[0.5, 1.5]$. We are interested in synthesizing a plan of length 4 (values of t_1, t_2, t_3, t_4) according to which the batteries will service the load for at least $t = 5$ before either of them dies. For this experiment we used $\epsilon_N = 0.01$ (ϵ can be ignored as the model features nondeterministic parameters only). Our algorithm explored 2,385 plans and found 63 of them such that for all the values in the obtained ranges neither battery dies within 5 seconds. For example, one of the plans is $t_1 \in [1.25, 1.375]$, $t_2 \in [1.25, 1.375]$, $t_3 \in [0.75, 0.875]$, $t_4 \in [0.875, 1]$. The computation took 133 min. on a 2.9GHz, 32-core system.

6 Related Work

Recently, the planning community has been quite active on hybrid domains, although without uncertain parameters, and considering continuous linear dynamics only (Shin and Davis 2005; Coles and Coles 2014; Coles et al. 2012) or discretizing nonlinearities (Della Penna et al. 2009). In (Bogomolov et al. 2014; 2015) the authors map PDDL+ (Fox and Long 2006) domains to (linear) hybrid automata, and then utilize the SpaceEx model checker (Frehse et al. 2011). The paper (Bryce et al. 2015) focuses on nonlinear PDDL+ domains and encode them as reachability problems which are solved by the δ-complete decision procedure dReal (Gao, Kong, and Clarke 2013). With respect to probabilistic planning, (Younes et al. 2005) introduced a probabilistic version of PDDL, albeit it focuses on discrete-state systems only. Also, efficient value iteration algorithms for Markov Decision Processes with imprecise probabilities have been presented in (Delgado et al. 2009).

For an overview of recent progress on SMT-based verification of (non-probabilistic) hybrid systems, see (Cimatti, Mover, and Tonetta 2012). An algorithm for deciding safety of robust (non-probabilistic) hybrid systems has been presented in (Ratschan 2014). SiSAT (Fränzle, Teige, and Eggers 2010) solves probabilistic bounded reachability using SMT, but it does not currently support continuous random parameters, and nondeterminism is solved by Monte Carlo simulation, so the approach can only provide statistical guarantees (Ellen, Gerwinn, and Fränzle 2015). In (Enszer and Stadtherr 2010) the authors present a technique for computing p-boxes for ODEs, but for finite-support random parameters and without giving theoretical guarantees. UPPAAL SMC (David et al. 2015) handles dynamic networks of stochastic hybrid automata, but it utilizes Monte Carlo simulation for nonlinear models. The PRISM model checker (Kwiatkowska, Norman, and Parker 2011) is limited to probabilistic timed automata. ProHVer computes an upper bound for the maximal reachability probability (Zhang et al. 2010), but handles continuous random parameters via discrete overapproximation (Fränzle et al. 2011). FAUST^2 (Soudjani, Gevaerts, and Abate 2015) utilizes abstraction to verify nondeterministic continuous-state, but discrete-time, Markov models.

7 Conclusions

We have presented an SMT-based approach for computing bounded reachability probability in uncertain hybrid automata. Uncertainty is modeled by parameters that can be random (continuous/discrete) or continuous nondeterministic. We do not pose any restriction on the model dynamics, and our approach can be applied to nonlinear hybrid domains. Our technique computes enclosures (*i.e.*, intervals) that are guaranteed to contain the reachability probabilities. Furthermore, we show that in some cases we can compute arbitrarily small enclosures. We have applied our technique to two examples of hybrid domains, and validated the results by Monte Carlo simulation. The experiments show that our technique is usable in practice, even for complex dynamics (*e.g.*, nonlinear differential equations). We have also showed that our algorithm can be used for synthesizing plans that are formally guaranteed to reach their goal. In the future we plan to generalize our technique to hybrid models featuring state-dependent probabilistic jumps and continuous probabilistic dynamics (stochastic differential equations).

Acknowledgement This work has been supported by award N00014-13-1-0090 of the US Office of Naval Research.

References

Alur, R., and Dill, D. L. 1990. Automata for modeling real-time systems. In *ICALP*, volume 443 of *LNCS*, 322–335.

Biere, A.; Cimatti, A.; Clarke, E. M.; and Zhu, Y. 1999. Symbolic model checking without BDDs. In *TACAS*, volume 1579 of *LNCS*, 193–207.

Bogomolov, S.; Magazzeni, D.; Podelski, A.; and Wehrle, M. 2014. Planning as model checking in hybrid domains. In *AAAI*, 2228–2234.

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ planning with hybrid automata: Foundations of translating must behavior. In *ICAPS*, 42–46.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. SMT-based nonlinear PDDL+ planning. In *AAAI*, 3247–3253.

Cimatti, A.; Mover, S.; and Tonetta, S. 2012. SMT-based verification of hybrid systems. In *AAAI*, 2100–2105.

Coles, A. J., and Coles, A. I. 2014. PDDL+ planning with events and linear processes. In *AAAI*, 74–82.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2012. COLIN: planning with continuous linear numeric change. *J. Artif. Intell. Res. (JAIR)* 44:1–96.

David, A.; Larsen, K.; Legay, A.; Mikučionis, M.; and Poulsen, D. B. 2015. Uppaal SMC tutorial. *International Journal on Software Tools for Technology Transfer (STTT)* 17(4):397–415.

Delgado, K. V.; Sanner, S.; de Barros, L. N.; and Cozman, F. G. 2009. Efficient solutions to factored MDPs with imprecise transition probabilities. In *ICAPS*, 98–105.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A tool for universal planning on PDDL+ problems. In *ICAPS*, 106–113.

Ellen, C.; Gerwinn, S.; and Fränzle, M. 2015. Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains. *International Journal on Software Tools for Technology Transfer (STTT)* 17(4):485–504.

Enszer, J. A., and Stadtherr, M. A. 2010. Verified solution and propagation of uncertainty in physiological models. *Reliable Computing* 15:168–178.

Fox, M., and Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *J. Artif. Intell. Res. (JAIR)* 27:235–297.

Fox, M.; Long, D.; and Magazzeni, D. 2012. Plan-based policies for efficient multiple battery load management. *J. Artif. Intell. Res. (JAIR)* 44:335–382.

Fränzle, M.; Hahn, E. M.; Hermanns, H.; Wolovick, N.; and Zhang, L. 2011. Measurability and safety verification for stochastic hybrid systems. In *HSCC*, 43–52.

Fränzle, M.; Teige, T.; and Eggers, A. 2010. Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. *J. Log. Algebr. Program.* 79(7):436–466.

Frehse, G.; Guernic, C. L.; Donzé, A.; Cotton, S.; Ray, R.; Lebeltel, O.; Ripado, R.; Girard, A.; Dang, T.; and Maler, O. 2011. SpaceEx: Scalable verification of hybrid systems. In *CAV*, volume 6806 of *LNCS*, 379–395.

Gao, S.; Avigad, J.; and Clarke, E. M. 2012. Delta-decidability over the reals. In *LICS*, 305–314.

Gao, S.; Kong, S.; Chen, W.; and Clarke, E. M. 2014. Delta-complete analysis for bounded reachability of hybrid systems. *CoRR* abs/1404.7171.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT solver for nonlinear theories over the reals. In *CADE-24*, volume 7898 of *LNCS*, 208–214.

Gonnet, P. 2012. A review of error estimation in adaptive quadrature. *ACM Comput. Surv.* 44(4):22:1–22:36.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *J. Amer. Statist. Assoc.* 58(301):13–30.

Ko, K.-I. 1991. *Complexity Theory of Real Functions*. Birkhäuser.

Kong, S.; Gao, S.; Chen, W.; and Clarke, E. M. 2015. dReach: Delta-reachability analysis for hybrid systems. In *TACAS*. *to appear.*

Kwiatkowska, M.; Norman, G.; and Parker, D. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, 585–591.

Petras, K. 2007. Principles of verified numerical integration. *Journal of Computational and Applied Mathematics* 199(2):317 – 328.

Ratschan, S. 2014. Safety verification of non-linear hybrid systems is quasi-decidable. *Formal Methods in System Design* 44(1):71–90.

Shin, J.-A., and Davis, E. 2005. Processes and continuous change in a SAT-based planner. *Artificial Intelligence* 166(12):194 – 253.

Soudjani, S. E. Z.; Gevaerts, C.; and Abate, A. 2015. FAUST^2: Formal abstractions of uncountable-state stochastic processes. In *TACAS*, volume 9035 of *LNCS*, 272–286.

Younes, H. L. S.; Littman, M. L.; Weissman, D.; and Asmuth, J. 2005. The first probabilistic track of the international planning competition. *J. Artif. Intell. Res. (JAIR)* 24:851–887.

Zhang, L.; She, Z.; Ratschan, S.; Hermanns, H.; and Hahn, E. M. 2010. Safety verification for probabilistic hybrid systems. In *CAV*, volume 6174 of *LNCS*, 196–211.