Counterexample Guided Abstraction Refinement for Polyhedral Probabilistic Hybrid Systems

RATAN LAL and PAVITHRA PRABHAKAR, Kansas State University, USA

We consider the problem of safety analysis of probabilistic hybrid systems, which capture discrete, continuous and probabilistic behaviors. We present a novel counterexample guided abstraction refinement (CEGAR) algorithm for a subclass of probabilistic hybrid systems, called polyhedral probabilistic hybrid systems (PHS), where the continuous dynamics is specified using a polyhedral set within which the derivatives of the continuous executions lie. Developing a CEGAR algorithm for PHS is complex owing to the branching behavior due to the probabilistic transitions, and the infinite state space due to the real-valued variables. We present a practical algorithm by choosing a succinct representation for counterexamples, an efficient validation algorithm and a constructive method for refinement that ensures progress towards the elimination of a spurious abstract counterexample. The technical details for refinement are non-trivial since there are no clear disjoint sets for separation. We have implemented our algorithm in a Python toolbox called Procegar; our experimental analysis demonstrates the benefits of our method in terms of successful verification results, as well as bug finding.

CCS Concepts: • **General and reference** → **Verification**; • **Theory of computation** → **Abstraction**;

Additional Key Words and Phrases: Probabilistic hybrid systems, safety, counterexample guided abstraction refinement

ACM Reference format:
Ratan Lal and Pavithra Prabhakar. 2019. Counterexample Guided Abstraction Refinement for Polyhedral Probabilistic Hybrid Systems. *ACM Trans. Embed. Comput. Syst.* 18, 5s, Article 98 (October 2019), 23 pages.
https://doi.org/10.1145/3358217

1 INTRODUCTION

Embedded control systems consist of physical systems controlled by digital software and often operate in uncertain environments, as in autonomous vehicles that need to navigate in complex scenarios involving other vehicles as well as pedestrians. Formal modeling and safety analysis of such systems has gained momentum in recent years owing to the safety criticality of the environments in which they are deployed. A formal model should capture the continuous behaviors of the physical system, the discrete behaviors of the digital controller, and the probabilistic aspects of the uncertain environment. Probabilistic/Stochastic Hybrid Systems [30, 37] provide one such formalism which extends a finite state automaton with probabilistic transitions and continuous

This article appears as part of the ESWEEK-TECS special issue and was presented at the International Conference on Embedded Software (EMSOFT) 2019.
Authors' addresses: R. Lal and P. Prabhakar, Kansas State University, USA; emails: {ratan, pprabhakar}@ksu.edu.

1539-9087/2019/10-ART98 $15.00
https://doi.org/10.1145/3358217

(stochastic) differential equations. In this paper, we focus on a subclass called the polyhedral probabilistic hybrid systems, where the continuous dynamics is captured using polyhedral differential inclusions of the form $\dot{x} \in P$, where $\dot{x}$ denotes the derivate of the state variable x with respect to time and P specifies a polyhedral set of rate vectors. We focus on this class of dynamics, since, they are reasonably amenable to analysis and can arbitrarily over-approximate more complex dynamics [34].

Finite state probabilistic systems such as discrete-time Markov chains (DTMCs) and Markov decision processes (MDPs) admit efficient verification algorithms [37]. Verification of hybrid systems (that capture discrete and continuous behaviors) is more challenging due to the infinite state-space, and is known to be undecidable for a relatively simple class of systems [23]. Efficient techniques based on finite state abstractions as well as symbolic exploration based on flowpipes have been explored [6, 11, 17, 18, 42]. Abstraction techniques have been extended to the setting of probabilistic hybrid systems [1, 45], however, finding right abstractions that result in successful verification remains a challenge. Techniques such as counterexample guided abstraction refinement (CEGAR) have proven to be useful in the context of finite state probabilistic systems as well as hybrid systems towards finding small abstractions that can prove safety of systems [7, 12, 26, 35, 36]. In this paper, we present a CEGAR algorithm for safety verification of polyhedral probabilistic hybrid systems. The broad approach consists of starting with abstraction and iteratively refining the abstraction based on a counterexample in the the abstract system that points to a potential violation of the property. To the best of our knowledge, this is the first CEGAR algorithm for probabilistic hybrid systems.

Developing a CEGAR algorithm for PHSs is challenging because it needs to address the branching behavior due to probabilities and the infinite state space due to continuous variables simultaneously. We view a given concrete PHS $\mathcal{H}$ as an infinite state MDP and start by constructing an abstract finite state MDP $\hat{\mathcal{H}}$ that simulates $\mathcal{H}$. We use the model-checker PRISM to obtain a counterexample of the abstract MDP, which is a DTMC in the form of a layered directed acyclic graph (LDAG), that is obtained by a finite unrolling of the DTMC returned by PRISM. Unlike DTMC, the LDAG has bounded paths and is more succinct than a tree unrolling. While the validation problem is a bounded model-checking problem, an SMT based approach for validation typically requires an exponential number of variables in the length of the counterexample [31]. We present an efficient validation procedure that performs a bottom-up backward reachability analysis and either finds a concrete counterexample or points to a node in the LDAG that needs to be refined. The refinement procedure is more involved as compared to the non-probabilistic setting due to the absence of clear disjoint sets that can be separated in the refinement step. Let us say that the validation fails at the layer k from the top at a node v, which corresponds to a set of concrete states S_v. Let $R_1, R_2, \ldots, R_n$ be the nodes reached using the backward reachable set computation at layer $k+1$. We know that the set of predecessors of the states $R_1, R_2, \ldots, R_n$ which are in S_v is empty (because the validation failed). This implies that the successors of the states S_v which correspond to a set of n tuples, say P_v, is disjoint from $R_1 \times R_2 \times \cdots \times R_n$. However, this does not immediately imply that the projection of P_v to the i-th component, namely, P_v^i is disjoint from R_i. Refinement in the purely hybrid setting is based on finding two disjoint sets, and splitting the corresponding node in the abstract system to separate a post set and a reach set. We observe that in general, we do not find two disjoint sets P_v^i and R_i for every i, nevertheless, we can split in a manner that ensures progress, that is, guarantees the elimination of the abstract counterexample in a finite number of refinements.

We implemented our algorithm in a Python toolbox called Procegar, and we compare our method with the tool ProHVer [45]. Our method concludes safety in many more instances than ProHVer. The method can return concrete counterexamples and it scales to higher dimensions than ProHVer.

The main contribution of the paper are

- We present the first CEGAR algorithm for the safety verification of probabilistic hybrid systems, to the best of our knowledge. This is achieved by defining a succinct notion of counterexample, an efficient validation algorithm and a non-trivial refinement algorithm that ensure progress.
- The experimental results suggest that the CEGAR method has benefits in comparison to existing abstraction based techniques such as the one implemented in ProHVer [45], due to the automatic refinement of abstractions.

2 RELATED WORKS

Stochastic hybrid systems [9, 25, 33] exhibit a combination of continuous and discrete behaviors where randomness is involved in both the behaviors. The probabilistic safety problem is undecidable in general, but it is decidable for certain subclasses with rectangular and o-minimal dynamics [40].

The problem has been studied for different subclasses such as Discrete-Time Markov Chain (DTMC) [20, 27], Continuous-Time Markov Chain (CTMC) [3, 21], and Markov Decision Process (MDP) [10, 29]. Our focus is on the probabilistic safety problem for stochastic hybrid systems. Different methods have been investigated for stochastic hybrid systems such as those based on finite abstraction [1, 15, 32, 45], approximate bisimulation [2, 44], bounded model checking [16, 43] based on dReach [28], statistical model checking [39] based on dReal [19], and stochastic satisfiability modulo theory based method [41].

While CEGAR techniques have been proposed in the context of finite state probabilistic systems [4, 10] and non-probabilistic hybrid systems [5, 35, 36], to the best of our knowledge, ours is the first CEGAR algorithm for models that have both probabilities and continuous dynamics, that is, probabilistic hybrid systems.

3 PRELIMINARIES

Numbers, Tuples, Sets, Functions. We use $Proj(t, j)$ to express the j^{th} projection of a tuple $t = (x_1, x_2, \ldots, x_n)$, that is, $Proj(t, j) = x_j$. Also, given an element q and a set $\mathcal{S}$, we use $(q, \mathcal{S})$ for $\{q\} \times \mathcal{S}$ and $(\mathcal{S}, q)$ for $\mathcal{S} \times \{q\}$. Given a set $\mathcal{S}$, we use $\mathcal{S}_{\mathbb{N}}$ to denote the set $\mathcal{S} \times \mathbb{N}$. Furthermore, relation $Id \subseteq \mathbb{R}^n \times \mathbb{R}^n$ expresses the identity relation. $\mathfrak{I}(f)$ denotes the set of restricted reverse onto functions of f as $\mathfrak{I}(f) = \{f_1 : B \to A \mid B \subseteq \mathcal{S}_2,\ A \subseteq \mathcal{S}_1,\ f_1$ is an onto function and $f o f_1 = Id\}$.

Polyhedra, Probability Distributions. We use $Poly(n)$ to denote the set of all polyhedra which are subsets of $\mathbb{R}^n$. Given a polyhedral set $\mathcal{S}$, we use $Cons(\mathcal{S})$ to denote a set of linear constraints expressing $\mathcal{S}$. We use $[\![c]\!]$ to denote the set of all points expressed by the linear constraint c. In addition, $Dist(\mathcal{S})$ denotes the set of all probability distributions over the set $\mathcal{S}$, that is, $Dist(\mathcal{S}) = \{\pi : \mathcal{S} \to [0, 1] \mid \sum_{s \in \mathcal{S}} \pi(s) = 1\}$. We use $support(\pi)$ to denote the set of elements $s \in \mathcal{S}$ for which $\pi(s) \neq 0$.

4 MARKOV DECISION PROCESS

In this section, we recall Markov Decision Processes and its subclass Discrete-Time Markov Chains.

Definition 4.1 (Markov Decision Process (MDP)*).* An MDP is a structure $\mathcal{M} = (\mathcal{S}, \longrightarrow)$, where

- ⋆ $\mathcal{S}$ is a finite (infinite) set of states;
- ⋆ $\longrightarrow \subseteq \mathcal{S} \times Dist(\mathcal{S})$ is a transition relation.

An MDP $\mathcal{M}$ consists of a set of states and zero or more transition probability distributions associated with each state. A probability distribution ρ associated with a state s, that is, $(s, \rho) \in \longrightarrow$, specifies the probability with which the system transitions from state s to each of the states s', that is, $\rho(s')$. A probabilistic edge $(s, \rho) \in \longrightarrow$ will also be denoted as $s \longrightarrow \rho$. It is possible that a state has multiple probabilistic edges, that is, $(s, \rho_1),\ (s, \rho_2) \in \longrightarrow$, where $\rho_1 \neq \rho_2$. Thus, it captures nondeterminism. We will use subscripts to refer to the components of a system, for instance, $\mathcal{S}_{\mathcal{M}}$ will refer to the states of $\mathcal{M}$ and $\longrightarrow_{\mathcal{M}}$ the transition relation of $\mathcal{M}$.

We define a finite (infinite) path of MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$ as a sequence of states $\sigma = s_0 s_1 s_2 \ldots$ such that there exist finite (infinite) sequence of probability distributions $\rho_0 \rho_1 \rho_1 \ldots$ which satisfy $s_i \longrightarrow \rho_i$ and $\rho_i(s_{i+1}) > 0$ for all $i \geq 0$. For a finite path $\sigma = s_0 s_1 s_2 \ldots s_n$, $\sigma[i]$ denotes the i^{th} state of the path σ, namely s_i, and $len(\sigma)$ represents the length of the path σ, that is, $len(\sigma) = n$. $Paths(\mathcal{M})$ denotes the set of all paths in $\mathcal{M}$. We say a state t is reachable from a state s if there exists a path $\sigma \in Paths(\mathcal{M})$ such that $\sigma[0] = s, \sigma[len(\sigma)] = t$. Given a set of states $\mathscr{F}$, $Paths(\mathcal{M}, s, \mathscr{F})$ represents the set of all paths $\sigma \in Paths(\mathcal{M})$ such that $\sigma[0] = s, \sigma[len(\sigma)] \in \mathscr{F}$ and there is no $i, 0 \leq i < len(\sigma)$ satisfying $\sigma[i] \in \mathscr{F}$. Next, we define discrete time Markov chains, which are MDPs, with at most one probabilistic edge associated with each state.

Definition 4.2 (Discrete-Time Markov Chain (DTMC)*).* A DTMC is an MDP $\mathcal{T} = (\mathcal{S}, \longrightarrow)$ such that for each state $s \in \mathcal{S}$, if there exist $s \longrightarrow \rho_1$ and $s \longrightarrow \rho_2$, then $\rho_1 = \rho_2$.

An execution of an MDP can be interpreted as a DTMC that is obtained by resolving the nondeterminism in each step with a particular state distribution. While a standard unrolling gives rise to a tree, here, we choose a compact representation of it in terms of a layered directed acyclic graph, wherein we do not track multiple copies of a state of the MDP at a particular level separately.

*Definition 4.3 (Layered Directed Acyclic Graph (*LDAG*)).* An LDAG $\mathcal{D} = (\mathcal{S}', \longrightarrow')$ is a special case of DTMC where $\mathcal{S}' \subseteq \mathcal{S} \times \mathbb{N}$ for some set $\mathcal{S}$ such that

- ⋆ There is a unique state $s \in \mathcal{S}$ such that $(s, 0) \in \mathcal{S}'$;
- ⋆ If $((s, i), \rho) \in \longrightarrow'$, then for all $t \in \mathcal{S}$, $\rho(t, j) = 0$ for $j \neq i + 1$;
- ⋆ If $(s, i) \in \mathcal{S}'$, then there exists $(s', i - 1) \longrightarrow' \rho$ for some $(s', i - 1) \in \mathcal{S}'$ such that $\rho(s, i) \neq 0$.

The LDAG begins with a unique root state at layer 0. All state transitions with positive probability correspond to jumps to states in the next layer. The last condition ensures that each state in the LDAG can be reached from the root. The depth d of a finite LDAG $\mathcal{D} = (\mathcal{S}', \longrightarrow')$ is the maximum length of any path in $\mathcal{D}$, that is, $d = \max\{i \mid (s, i) \in \mathcal{S}'\}$. Next, we define a sub-LDAG as a substructure of an LDAG rooted at a certain state/node.

Definition 4.4 (Sub-LDAG). Given an LDAG $\mathcal{D} = (\mathcal{S}, \longrightarrow)$, a sub-LDAG $\mathcal{D}' = (\mathcal{S}', \longrightarrow')$ rooted at $(s, j) \in \mathcal{S}$ is defined as

- ⋆ $\mathcal{S}' = \{(s', i - j) \mid (s', i)$ is reachable from (s, j) in $\mathcal{D}\}$;
- ⋆ $\longrightarrow' = \{((s', i), \rho') \mid (s', i) \in S', \exists((s', i + j), \rho) \in \longrightarrow, \forall(s'', i + 1) \in \mathcal{S}', \rho'((s'', i + 1)) = \rho(s'', i + j + 1)\}$

We define executions of $\mathcal{M}$ as LDAGs that arise due to unrolling of $\mathcal{M}$.

Definition 4.5. Given an MDP $(\mathcal{S}, \longrightarrow)$, $Exec(\mathcal{M})$ is the set of all LDAGs $\mathcal{D} = (\mathcal{S}', \longrightarrow')$ where $\mathcal{S}' \subseteq \mathcal{S} \times \mathbb{N}$; and for every $((s, i), \rho') \in \longrightarrow'$, then there is ρ such that $(s, \rho) \in \longrightarrow$ and for every $(t, i + 1) \in \mathcal{S}'$, $\rho'(t, i + 1) = \rho(t)$.

Next, we associate probabilities with paths, the probability associated with a path σ is denoted by $Prob(\sigma)$.

- For DTMC $\mathcal{T} = (\mathcal{S}, \longrightarrow)$, for path σ, if $len(\sigma) = 0$, then $Prob(\sigma) = 1$. If $len(\sigma) > 0$, then

$$Prob(\sigma) = \prod_{i=0}^{len(\sigma)-1} \rho_i(\sigma[i+1]), \text{ where } (\sigma[i], \rho_i) \in \longrightarrow .$$

  The probability of reaching a set of states $\mathscr{F}$ from a state s is defined as,

$$Prob(\mathcal{T}, s, \mathscr{F}) = \sum_{\sigma \in Paths(\mathcal{T}, s, \mathscr{F})} Prob(\sigma).$$

- For MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$, we define the minimum and maximum probability of reaching a set of states $\mathscr{F}$ from a state s, denoted as $Prob_{\min}(\mathcal{M}, s, \mathscr{F})$ and $Prob_{\max}(\mathcal{M}, s, \mathscr{F})$, respectively. Let $\bowtie \in \{\min, \max\}$.

$$Prob_{\bowtie}(\mathcal{M}, s, \mathscr{F}) = \bowtie_{\mathcal{D} \in Exec(\mathcal{M})} Prob(\mathcal{D}, (s, 0), \mathscr{F} \times \mathbb{N}).$$

5 PROBABILISTIC SIMULATION

Probabilistic simulation captures a relation between states of two MDPs wherein if two states are related then any probabilistic transition from the simulated state can be mimicked by the simulating state. In this section, we consider a restricted version of the probabilistic simulation relation between two probabilistic systems as defined in [38], which is sufficient to capture the relation between the concrete system and its abstraction defined in this paper. In particular, we enforce a bijection between the probability distribution associated with the two states. We consider this restricted version because it simplifies our technical discussion and suffices for our purposes.

Definition 5.1. Let $\mathcal{M}_1 = (\mathcal{S}_1, \longrightarrow_1)$ and $\mathcal{M}_2 = (\mathcal{S}_2, \longrightarrow_2)$ be two MDPs. An onto function $\alpha : \mathcal{S}_1 \to \mathcal{S}_2$ is a probabilistic simulation from $\mathcal{S}_1$ to $\mathcal{S}_2$ if for all $s_1 \in \mathcal{S}_1$, $s_2 \in \mathcal{S}_2$ such that $\alpha(s_1) = s_2$ and $s_1 \longrightarrow \pi$, $\exists\, \rho \in Dist(\mathcal{S}_2)$ such that $s_2 \longrightarrow \rho$, and

- ⋆ $\pi(s) = \rho(\alpha(s))$, for all $s \in support(\pi)$;
- ⋆ For $s, s' \in support(\pi)$ if $s \neq s'$, then $\alpha(s) \neq \alpha(s')$.

We say that α is a probabilistic simulation from $\mathcal{M}_1$ to $\mathcal{M}_2$, denoted by $\mathcal{M}_1 \leq_\alpha \mathcal{M}_2$. In addition, given $\mathscr{F} \subseteq \mathcal{S}_1$, we say that α respects $\mathscr{F}$ if for $s \in \mathscr{F}$ and $t \in \mathcal{S}_1 \setminus \mathscr{F}$, $\alpha(s) \neq \alpha(t)$.

Next, the following theorem states the fact that if two MDPs $\mathcal{M}_1$ and $\mathcal{M}_2$ are related by a probabilistic simulation α, that is, $\mathcal{M}_1 \leq_\alpha \mathcal{M}_2$, then MDP $\mathcal{M}_2$ is an over-approximation of $\mathcal{M}_1$, that is, $\mathcal{M}_2$ has a larger number of executions than $\mathcal{M}_1$; thus the minimum and maximum probability of reaching a set of states in $\mathcal{M}_1$ is lower bounded by the minimum and upper bounded by the maximum probability of reaching the related set of states in $\mathcal{M}_2$. The following theorem is similar to the theorem given in the paper [14].

Theorem 5.2. *Let $\mathcal{M}_1 = (\mathcal{S}_1, \longrightarrow_1)$ and $\mathcal{M}_2 = (\mathcal{S}_2, \longrightarrow_2)$ be two* MDP*s and $\mathscr{F}_1 \subseteq \mathcal{S}_1$. If α is a probabilistic simulation from $\mathcal{M}_1$ to $\mathcal{M}_2$ which respects $\mathscr{F}_1$ then for any $s_1 \in \mathcal{S}_1$, we have*

(a) $Prob_{\max}(\mathcal{M}_1, s_1, \mathscr{F}_1) \leq Prob_{\max}(\mathcal{M}_2, \alpha(s_1), \alpha(\mathscr{F}_1))$;
(b) $Prob_{\min}(\mathcal{M}_2, \alpha(s_1), \alpha(\mathscr{F}_1)) \leq Prob_{\min}(\mathcal{M}_1, s_1, \mathscr{F}_1)$.

Next, we define a function $\alpha_{\mathbb{N}}$ that extends probabilistic simulation relation between states of MDPs to states of corresponding executions.

Definition 5.3. Given a probabilistic simulation $\alpha : \mathcal{S}_1 \to \mathcal{S}_2$ between two MDPs $\mathcal{M}_1 = (\mathcal{S}_1, \longrightarrow_1)$ and $\mathcal{M}_2 = (\mathcal{S}_2, \longrightarrow_2)$, we define $\alpha_{\mathbb{N}} : \mathcal{S}_1 \times \mathbb{N} \to \mathcal{S}_2 \times \mathbb{N}$ such that $\alpha_{\mathbb{N}}(s, i) = (\alpha(s), i)$ for each $s \in \mathcal{S}_1$, $i \in \mathbb{N}$.

Furthermore, we define the set of all executions $\mathcal{D}_1$ of an MDP $\mathcal{M}_1$ which simulate an execution $\mathcal{D}_2$ of an MDP $\mathcal{M}_2$ with respect to the probabilistic simulation α between $\mathcal{M}_1$ and $\mathcal{M}_2$.

Definition 5.4. Given a probabilistic simulation $\alpha : \mathcal{S}_1 \to \mathcal{S}_2$ between two MDPs $\mathcal{M}_1 = (\mathcal{S}_1, \longrightarrow_1)$ and $\mathcal{M}_2 = (\mathcal{S}_2, \longrightarrow_2)$, an LDAG $\mathcal{D}_2 \in \mathit{Exec}(\mathcal{M}_2)$, we define $\alpha^{-1}(\mathcal{D}_2)$ as follow:

$$\alpha^{-1}(\mathcal{D}_2) = \{\mathcal{D}_1 \in \mathit{Exec}(\mathcal{M}_1) \mid \mathcal{D}_2 \leq_\gamma \mathcal{D}_1 \text{ for some } \gamma \in \Im(\alpha_{\mathbb{N}})\}.$$

Additionally, we define an abstraction of an MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$ denoted $\alpha(\mathcal{M})$ with respect to an abstraction function $\alpha : \mathcal{S} \to \mathcal{S}'$ for some $\mathcal{S}'$. The abstraction function is formally defined as follows.

Definition 5.5. Given an MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$, an abstraction function $\alpha : \mathcal{S} \to \mathcal{S}'$ for some set $\mathcal{S}'$ is a function such that for any $(s, \rho) \in \longrightarrow$, if $s_1, s_2 \in \mathit{support}(\rho)$, then $\alpha(s_1) \neq \alpha(s_2)$, where $s_1 \neq s_2$.

The above definition appears to be a restricted notion of abstraction on the MDPs, however, the particular abstractions we consider on probabilistic hybrid systems in the sequel will correspond to abstraction functions of this kind on the underlying MDPs. Furthermore, the formal definition of $\alpha(\mathcal{M})$ is given as below.

Definition 5.6. Given an MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$ and an abstraction function $\alpha : \mathcal{S} \to \mathcal{S}'$, $\alpha(\mathcal{M}) = (\mathcal{S}', \longrightarrow')$, where $\longrightarrow' = \{(s', \rho') \mid \exists (s, \rho) \in \longrightarrow, \alpha(s) = s', \forall t, \rho(t) = \rho'(\alpha(t))\}$.

The theorem below states that $\alpha(\mathcal{M})$ is an over-approximation of $\mathcal{M}$.

Theorem 5.7. *Given an* MDP $\mathcal{M} = (\mathcal{S}, \longrightarrow)$*, an abstraction function* $\alpha : \mathcal{S} \to \mathcal{S}'$ *for some set* $\mathcal{S}'$*, we have* $\mathcal{M} \leq_\alpha \alpha(\mathcal{M})$.

6 PROBABILISTIC HYBRID SYSTEMS

In this paper, we study polyhedral probabilistic hybrid systems that capture the discrete, continuous and probabilistic behaviors. The continuous behaviors are captured by specifying a polyhedral inclusion dynamics, and guards, resets and probabilistic edges capture the discrete and probabilistic behaviors. Next, we introduce the formal definition of polyhedral probabilistic hybrid systems.

*Definition 6.1 (Polyhedral Probabilistic Hybrid Systems (*PHS*)).* A polyhedral probabilistic hybrid system is a tuple $\mathcal{H} = (Q, \mathcal{X}, \mathit{Inv}, \mathit{Flow}, \mathit{Edges}, \mathit{Guard}, \mathit{Reset})$ where

- ⋆ Q is a finite set of locations;
- ⋆ $\mathcal{X} \subseteq \mathbb{R}^n$ is a continuous statespace;
- ⋆ Inv: $Q \to \mathit{Poly}(n)$ is an invariant function;
- ⋆ Flow: $Q \to \mathit{Poly}(n)$ is a flow function;
- ⋆ $\mathit{Edges} \subseteq Q \times \mathit{Dist}(Q)$ is a finite set of probabilistic edges;
- ⋆ Guard: $\mathit{Edges} \to \mathit{Poly}(n)$ is a guard function;
- ⋆ Reset: $\mathit{Edges} \times Q \to \mathit{Poly}(2n)$ is a reset function.

The instantaneous description of the system is captured using a state (q, x), where q is a discrete location, and x is a continuous state. The system evolves continuously from a state (q, x) for a time t by following a vector in $\mathit{Flow}(q)$ at all times and staying within $\mathit{Inv}(q)$, and eventually reaching a state (q, y) at time t. If the state reached satisfies the guard associated with an edge (q, ρ), that is, if $y \in \mathit{Guard}(q, \rho)$, then the system can take transition to a state (q', x') with probability $\rho(q')$, and the new continuous state x' is specified by the reset function, and satisfies $(y, x') \in \mathit{Reset}((q, \rho), q')$. The system does not necessarily have to take a transition if a guard is enabled, but for the system to take a transition, the corresponding guard needs to be enabled.

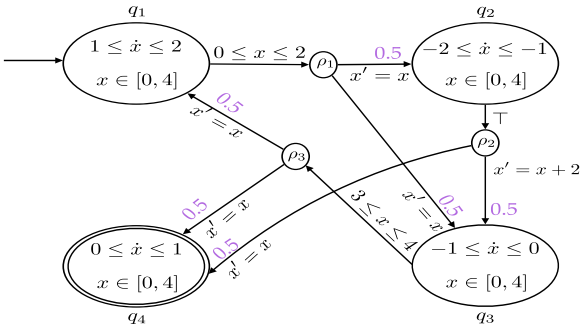


Fig. 1. Polyhedral probabilistic hybrid system.

Example 6.2. Figure 1 illustrates a polyhedral PHS, where the locations q_1, q_2, q_3 and q_4 are represented using circle nodes. The variable x represents the continuous variable which takes real values, that is, the continuous statespace is $\mathbb{R}$. Each location is annotated with a polyhedral inclusion dynamics and an invariant which are written within the corresponding circle. A polyhedral inclusion dynamics is represented using a constraint over the rates of the continuous state variables, and the invariant is a constraint over the continuous state variables which constrains the system evolution at a particular location. For example, in location q_1, the polyhedral inclusion dynamics is $1 \le \dot{x} \le 2$, where $\dot{x}$ is the rate of continuous state evolution, and the invariant is $x \in [0, 4]$. Next, the switching from one location to another location is expressed by a probabilistic edge which has mainly three components, namely, the guard, the probability distribution and the reset. The guard is a constraint over the continuous state variables that represents the condition that enables the system to switch between locations, the probability distribution provides the probability of switching to a target location, and the reset expressed as a constraint over the continuous state variables and its primed versions, provides the values of the continuous state before and after a transition, respectively. For example, in location q_1, the system has a probabilistic edge with guard $0 \le x \le 2$, probability distribution $\rho_1(q_2) = \rho_1(q_3) = 0.5$ and reset for locations q_2 and q_3 as $x' = x$. Hence, when the system is in location q_1 and the continuous state x satisfies the guard $0 \le x \le 2$, the system switches to locations q_2 and q_3 with probability 0.5 each, and the continuous state remains unchanged ($x_p = x$).

The semantics of the PHS is captured as an infinite state MDP as follows. We use primed version of location and continuous state variables to denote the location and continuous state after a continuous or discrete transition.

Definition 6.3. Given PHS $\mathcal{H} = (Q, \mathcal{X}, Inv, Flow, Edges, Guard, Reset)$, the semantics of $\mathcal{H}$ is defined as an Markov Decision Process $[\![\mathcal{H}]\!] = (\mathcal{S}_{[\![\mathcal{H}]\!]}, \longrightarrow_{[\![\mathcal{H}]\!]})$, where

- $\mathcal{S}_{[\![\mathcal{H}]\!]} = Q \times \mathcal{X}$;
- $((q, x), \pi) \in \longrightarrow_{[\![\mathcal{H}]\!]}$ if $\exists\ (q, \rho) \in Edges$, $\Phi : [0, T] \to \mathcal{X}$ such that
  - $\Phi(0) = x$, $\Phi(T) \in Guard(q, \rho)$;
  - for $0 \le t \le T$, $\Phi(t) \in Inv(q)$ and $\frac{d\Phi}{dt}(t) \in Flow(q)$;
  - for every $q' \in Q$, $\pi(q', x') = \rho(q')$ for some x' such that $(\Phi(T), x') \in Reset((q, \rho), q')$.

The MDP $[\![\mathcal{H}]\!]$ has an infinite number of states because each state is a pair of discrete location and continuous state, where the number of values of continuous states is infinite. Next, we formally define the probabilistic safety problem.

Problem 1 (The Maximum Probabilistic Safety Problem). *Given a polyhedral* PHS $\mathcal{H}$*, an initial location* q_0*, a set of final locations* $\mathbb{F}$ *and probability bound* $\theta \in [0,1]$*, check whether the maximum probability of reaching* $\mathbb{F} \times \mathcal{X}$ *from* $\{q_0\} \times Inv(q_0)$ *is less than or equal to* θ*, that is,* $\max_{s \in \{q_0\} \times Inv(q_0)} Prob_{\max}([\![\mathcal{H}]\!], s, \mathbb{F} \times \mathcal{X}) \leq \theta$.

Similarly, the minimum probabilistic reachability problem can be defined. Here, we focus on the maximum probabilistic safety problem. However, our results extend in a straightforward manner to verify lower bound on the minimum probability of reachability. In Figure 1, the initial location is indicated with an incoming arrow and the final location is indicated with double circles.

7 COUNTER EXAMPLE GUIDED ABSTRACTION REFINEMENT

In this section, we provide the details of our Counter-Example Guided Abstraction Refinement (CEGAR) [13] algorithm for the probabilistic safety analysis of polyhedral PHS. The maximum probabilistic safety problem for polyhedral PHS is challenging due to the infinite statespace. An abstraction based analysis consists of *abstracting* a given (concrete) system to a simpler abstract system, such that the satisfaction of a certain property, such as, the maximum probability of reaching an unsafe state being less than θ, on the abstract system, implies the satisfaction of the property on the concrete system. In our case, the concrete system is a polyhedral PHS and the abstract system is a finite state MDP, on which the maximum probabilistic safety analysis can be carried out efficiently. One of the challenges towards abstraction based analysis is the choice of the abstraction, on which the success of the verification crucially relies. In other words, the abstract system is often an "over-approximation" and hence, the violation of the property by the abstract system does not imply a violation in the concrete system. However, a violation in the abstract system provides a potential reason for the violation of the property in the concrete system, in the form of an *abstract counterexample* (ACE). Analysis of the abstract counterexample using a *validation* algorithm, results in either a concrete counterexample corresponding to it (a bug) or in concluding that the abstract counterexample is spurious, that is, does not correspond to a real counterexample. In the latter case, the spurious counterexample analysis can be used to *refine* the abstract system so as to eliminate the same. This process is continued until either some abstraction of the concrete system satisfies the property or the validation algorithm determines that the system violates the property. This iterative refinement of the abstract system based on counterexamples is referred to as the counterexample guided abstraction refinement. While CEGAR algorithms have been proposed for (non-probabilistic) hybrid systems [35] and finite state probabilistic systems [24], there are several challenges due to the combination of infinite statespace and the branching behavior in a probabilistic hybrid system. In particular, we need a succinct representation of the abstract counterexample, efficient validation and refinement algorithms. We provide the details of each of the steps of the CEGAR loop in the sequel.

7.1 Abstraction

In this section, we describe an abstraction procedure to abstract a polyhedral PHS into an MDP. The broad idea is to divide the statespace of each mode into a finite number of regions, and construct an MDP which consists of these regions as abstract nodes. Let us fix a polyhedral PHS $\mathcal{H} = (Q, \mathcal{X}, Inv, Flow, Edges, Guard, Reset)$ and a finite partition $\mathcal{R}$ of $Q \times \mathcal{X}$ such that for any partition element $\mathcal{P} \in \mathcal{R}$ if $(q,x), (q',x') \in \mathcal{P}$ then $q = q'$. So, $\mathcal{R}$ consists of a partitioning of the statespace where states from different modes are kept separate. Consider a function $\alpha_{\mathcal{R}} : Q \times \mathcal{X} \to \mathcal{R}$ such that $\alpha_{\mathcal{R}}(q,x) = (q,R)$, where $x \in R$. That is, $\alpha_{\mathcal{R}}$ maps a concrete state to the corresponding region in the partition it belongs to. Note that $\alpha_{\mathcal{R}}$ is an abstraction function, because, for any $((q,x),\rho) \in \longrightarrow_{[\![\mathcal{H}]\!]}$, if $(q_1,x_1),(q_2,x_2) \in support(\rho)$ and $(q_1,x_1) \neq (q_2,x_2)$, then $q_1 \neq q_2$ (from the

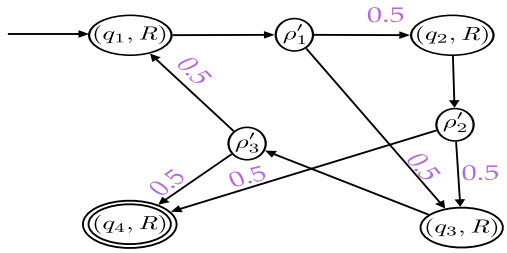


Fig. 2. Markov decision process $Abs(\mathcal{H}, \mathcal{R})$.

semantics, there is a unique successor state corresponding to every target location), and hence, $\alpha_{\mathcal{R}}(q_1, x_1) \neq \alpha_{\mathcal{R}}(q_2, x_2)$.

The abstraction of $\mathcal{H}$ with respect to the finite partition $\mathcal{R}$ denoted as $Abs(\mathcal{H}, \mathcal{R})$ is $\alpha_{\mathcal{R}}([\![\mathcal{H}]\!])$. From Theorem 5.7, $Abs(\mathcal{H}, \mathcal{R})$ is an over-approximation of polyhedral PHS $\mathcal{H}$. From Theorem 5.2, if the abstract system $Abs(\mathcal{H}, \mathcal{R})$ is probabilistically safe, that is, $Prob_{\max}(Abs(\mathcal{H}, \mathcal{R}), (q_0, R), \alpha_{\mathcal{R}}(\mathbb{F} \times \mathcal{X})) \leq \theta$ for every $(q_0, R) \in \alpha_{\mathcal{R}}(\{q_0\} \times Inv(q_0))$, then the concrete system $\mathcal{H}$ is also probabilistically safe, that is, $Prob_{\max}([\![\mathcal{H}]\!], (q_0, x), \mathbb{F} \times \mathcal{X}) \leq \theta$ for every $(q_0, x) \in \{q_0\} \times Inv(q_0)$.

If $\mathcal{R}$ is a polyhedral partition, then $Abs(\mathcal{H}, \mathcal{R})$ can be constructed. Note that from Definition 5.6, to check whether a probabilistic edge exists in the abstract system, we need to check if a corresponding concrete edge exists. This can be encoded as a satisfiability problem of a conjunction of linear constraints. Alternately, one could use polyhedral manipulations to compute the predecessor states corresponding to the abstract probabilistic edge in the concrete system, and check for its emptiness. The next example illustrates the computation of the abstract MDP.

Example 7.1. Consider the polyhedral PHS $\mathcal{H}$ with its state space $\mathcal{Q} \times \mathcal{X} = \{q_1, q_2, q_3, q_4\} \times [0, 4]$. Let $\mathcal{R} = \{(q_1, R), (q_2, R), (q_3, R), (q_4, R)\}$ be a finite partition of the state space $\mathcal{Q} \times \mathcal{X}$, where $R = [0, 4]$. The abstraction of $\mathcal{H}$ with respect to the partition $\mathcal{R}$ is $Abs(\mathcal{H}, R)$ which we construct according to Definition 5.6. Each partition element of $\mathcal{R}$ is an abstract state, for instance, (q_1, R) is a state of the abstract system as shown in Figure 2. Next, we construct a probabilistic edge from one abstract state to a tuple of abstract states with certain probabilities if there is a corresponding concrete edge in $\mathcal{H}$. For example, we construct an abstract edge from (q_1, R) to (q_2, R) and (q_3, R) with probabilities 0.5 and 0.5, respectively, because a corresponding concrete edge exists in $\mathcal{H}$, as given next. The system starts from $x = 0.5 \in R$ at location q_1 and evolves for time $t = 2$ with flow $\dot{x} = 0.5$ and reaches the state $x_1 = x + \dot{x}t = 0.5 + 0.5 * 2 = 1.5$ which satisfies the guard constraints, that is, $1.5 \in [0, 2]$. Then, the system switches to q_2 and q_3 by resetting the continuous states to $x_2 = x_1 = 1.5$ and $x_3 = x_1 = 1.5$, respectively. Since $x_2, x_3 \in R$, we have a concretization of the edge from (q_1, R) to (q_2, R) and (q_3, R) with probability 0.5 each.

7.2 Model Checking and Counter Example

In this section, we verify the maximum probabilistic safety problem for the abstract system $Abs(\mathcal{H}, \mathcal{R})$. Since the abstract system is a finite MDP, we adopt probabilistic model checker tool PRISM [30] for the purpose of verification. PRISM [30] takes as input a finite MDP, a set of initial states and a PCTL formula [22], and returns either *true* to indicate that the MDP satisfies the formula, or returns a counterexample $\widehat{\mathcal{T}}$ in the form of a DTMC. Maximum probabilistic safety specification can be encoded as a PCTL-formula. While $\widehat{\mathcal{T}}$ is a counterexample, it contains cycles and hence, analyzing it for spuriousness is complex. Hence, we unroll $\widehat{\mathcal{T}}$ and construct a finite LDAG $\widehat{\mathcal{D}}$ as a succinct counterexample. Next, we provide the formal definition of a counterexample.

Definition 7.2. Given an MDP $\mathcal{M}$, an initial state q_0, a set of final states $\mathbb{F}$ and $\theta \in [0,1]$, an LDAG $\widehat{\mathcal{D}} \in Exec(\mathcal{M})$ is a counterexample of $\mathcal{M}$ with respect to q_0, $\mathbb{F}$ and θ, if $Prob(\widehat{\mathcal{D}}, (q_0, 0), \mathbb{F} \times \mathbb{N}) > \theta$.

Algorithm 1 provides the details of extracting a succinct LDAG counterexample $\widehat{\mathcal{D}}$ from a given DTMC counterexample $\widehat{\mathcal{T}}$ that is returned by PRISM which indicates that the probability of reaching $\mathbb{F}$ in $\widehat{\mathcal{T}}$ from q_0 is greater than θ. The crux of the algorithm consist of unrolling the DTMC and collecting enough paths reaching $\mathbb{F}$ from q_0 whose sum total exceeds θ. The algorithm iteratively computes p_q^i, which is the probability of reaching a final state in $\mathbb{F}$ from q within i steps. It can be observed that there is always a d for which $p_{q_0}^d$ exceeds θ. To see this, note that in the DTMC $\widehat{\mathcal{T}}$, the probability of reaching $\mathbb{F}$ from q is $> \theta$, which is the sum of probabilities of paths reaching $\mathbb{F}$. Since, the DTMC is finite, we can enumerate the paths, and the corresponding probabilities. Let $p_1, p_2, \ldots$ be the probabilities of the paths reaching $\mathbb{F}$, where $\sum_i p_i > \theta$. Our objective is to show that there is finite number of these probabilities (correspond to finitely many paths) who sum exceeds θ. Note that if the sequence is finite, we can take all the paths that reach $\mathbb{F}$. Otherwise, $\sum_i p_i > \theta$ implies that $\lim_{i\to\infty} \sum_{j=1}^{i} p_i > \theta$ by definition of infinite sum. From the definition of limit, there is a k such that $\sum_{j=1}^{i} p_i > \theta$ for all $i \geq k$. In particular, $\sum_{j=1}^{k} p_i > \theta$. Hence, in at most k iterations, p_q^d will exceed θ and exit the loop. Once the length d for the sufficient unrolling is computed, the function ForwardReach performs a forward reachability analysis to only keep those nodes in the unrolling of $\mathcal{M}$ up to length d that can be reached from q_0 using transitions in the support (that is, edges with non-zero probability).

ALGORITHM 1: Get_LDAG: Extract a succinct LDAG counterexample from a given DTMC counterexample

Input: $\widehat{\mathcal{T}}$ - a DTMC, $\mathbb{F}$ - a set of final states, θ - a probability bound, q_0 - a state of $\widehat{\mathcal{T}}$

Output: a finite LDAG $\widehat{\mathcal{D}}$

```
1  begin
2      for each state q of T̂ do
3          if q ∈ F then p_q^0 = 1 else p_q^0 = 0
4      d = 0
5      while p_{q0}^d ≤ θ do
6          d = d+1
7          for each edge (q, ρ) of T̂ do
8              if q ∈ F then p_q^d = 1 else p_q^d = Σ_{state s' of T̂} ρ(s') * p_{s'}^{d-1}
9      D̂ = ForwardReach(M, q0, d) return D̂
10 end
```

Example 7.3. Consider the MDP $Abs(\mathcal{H}, \mathcal{R})$ shown in Figure 2, an initial state (q_1, R), a set of final states $\{(q_4, R)\}$ and $\theta = 0.5$. The LDAG $\widehat{\mathcal{D}} \in Exec(Abs(\mathcal{H}, \mathcal{R}))$ shown in Figure 3 is a counterexample, because $Prob(\widehat{\mathcal{D}}, ((q_1, R), 0), \{(q_4, R)\} \times \mathbb{N}) = Prob(\sigma_1) + Prob(\sigma_2) + Prob(\sigma_3) = 0.25 + 0.25 + 0.125 = 0.625$, which is greater than 0.50, where σ_1, σ_2 and σ_3 are paths given by $\sigma_1 = ((q_1, R), 0) \to ((q_2, R), 1) \to ((q_4, R), 2)$, $\sigma_2 = ((q_1, R), 0) \to ((q_3, R), 1) \to ((q_4, R), 2)$ and $\sigma_3 = ((q_1, R), 0) \to ((q_2, R), 1) \to ((q_3, R), 2) \to ((q_4, R), 3)$ in $\widehat{\mathcal{D}}$.

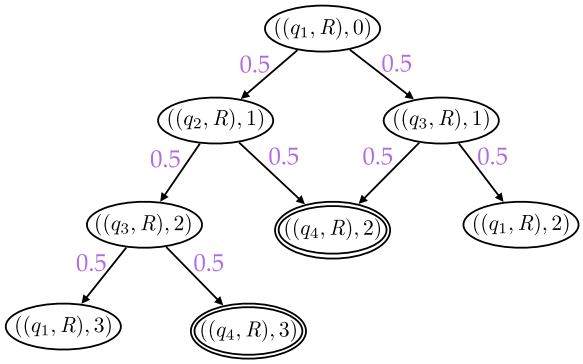


Fig. 3. LDAG $\widehat{\mathcal{D}} \in \mathit{Exec}(\mathit{Abs}(\mathcal{H}, \mathcal{R}))$ violating $\mathit{Prob}_{\max}$.

If $\mathit{Abs}(\mathcal{H}, \mathcal{R})$ is not probabilistically safe, then an abstract counterexample $\widehat{D}$ of $\mathit{Abs}(\mathcal{H}, \mathcal{R})$ can be computed. The abstract counterexample indicates only a potential violation of safety in the concrete system. Hence, we need to validate the counterexample $\widehat{D}$ to check if it is realizable in the concrete system $\mathcal{H}$. Next, we provide the details about the validation of a counterexample.

7.3 Validation

In this section, we provide a method to validate whether a counterexample expressed as an LDAG for the abstract system $\mathit{Abs}(\mathcal{H}, \mathcal{R})$ is realizable in the concrete system $\mathcal{H}$. Let us fix a finite LDAG $\widehat{\mathcal{D}} = (\mathcal{S}_{\widehat{\mathcal{D}}}, \longrightarrow_{\widehat{\mathcal{D}}})$ as a counter example of $\mathit{Abs}(\mathcal{H}, \mathcal{R})$ whose depth is d. Our validation problem is to check whether $\widehat{\mathcal{D}}$ is realizable in the concrete system $\mathcal{H}$. This is formally defined as below.

Problem 2 (Validation Problem). *Verify whether there exists* $\mathcal{D} \in \mathit{Exec}([\![\mathcal{H}]\!])$ *such that* $\mathcal{D} \in \alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$.

Our broad approach for the validation of $\widehat{\mathcal{D}}$ is to compute all those concrete set of states associated with the abstract states from which sub-LDAG of $\widehat{\mathcal{D}}$ rooted at the abstract state is realizable in $\mathcal{H}$. We compute such sets from bottom to top layer, which is based on backward reachability analysis for which we define a function *Pre*.

Definition 7.4. Given a set S, a finite set of sets $\{S_i\}_{i \in A}$ and $\{p_i\}_{i \in A}$ for some index set A, we define a function $\mathit{Pre}_{[\![\mathcal{H}]\!]}(S, \{S_i\}_{i \in A}, \{p_i\}_{i \in A})$ as

$$\{s \in S \mid \exists\, \rho,\ s \longrightarrow_{[\![\mathcal{H}]\!]} \rho,\ \forall\, i \in\ A,\ \exists\, s_i \in S_i \text{ such that } \rho(s_i) = p_i\}.$$

We are going to compute a set of concrete states $\Re^k_{(q,R)}$ for each abstract state $((q, R), k)$ of $\widehat{\mathcal{D}}$, where $\Re^k_{(q,R)}$ denotes the set of all concrete states $(q, x) \in \{q\} \times R$ from which there exists an LDAG $\mathcal{D} \in \mathit{Exec}([\![\mathcal{H}]\!])$ such that the sub-LDAG $\mathcal{D}'$ of $\widehat{\mathcal{D}}$ rooted at $((q, R), k)$ is realizable by $\mathcal{D}$ in the concrete system $\mathcal{H}$. This is formally defined as follows.

Definition 7.5. Given an abstract state $((q, R), k) \in \mathcal{S}_{\widehat{\mathcal{D}}}$, we have $\Re^k_{(q,R)}$ as $\{(q, x) \in \{q\} \times R \mid \exists\, \mathcal{D} \in \mathit{Exec}([\![\mathcal{H}]\!])$ rooted at $((q, x), 0)$ for which there exists a sub-LDAG $\mathcal{D}'$ of $\widehat{\mathcal{D}}$ rooted at $((\mathrm{q}, \mathrm{R}), k)$ such that $\mathcal{D} \in \alpha_{\mathcal{R}}^{-1}(\mathcal{D}')\}$.

Alternatively, $\Re^k_{(q,R)}$ can be inductively defined as follows. Let $\mathcal{S}^k_{(q,R)}$ be the set of all concrete states associated with the abstract state $((q, R), k)$, that is, $\mathcal{S}^k_{(q,R)} = \alpha_{\mathcal{R}}^{-1}(q, R)$, and Q_k be the set of all those abstract states (q, R) at layer k in $\widehat{\mathcal{D}}$, that is, $((q, R), k) \in \mathcal{S}_{\widehat{\mathcal{D}}}$.

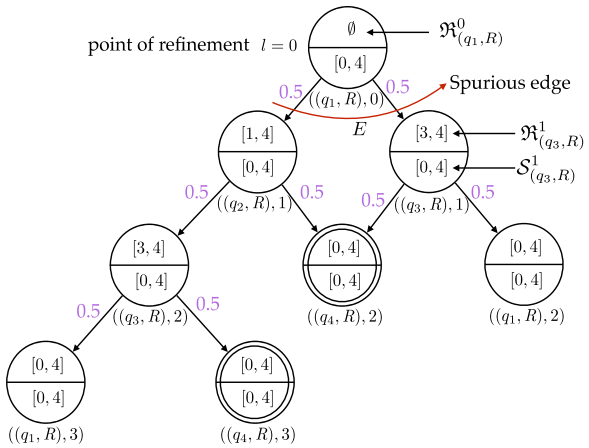


Fig. 4. Point of refinement and spurious edge in $\widehat{\mathcal{D}}$ shown in Figure 3.

- ⋆ If $k = d$ or there is no abstract edge $((q,R),k) \longrightarrow_{\widehat{\mathcal{D}}} \rho$, then $\mathfrak{R}^k_{(q,R)}$ is $\mathcal{S}^k_{(q,R)}$;
- ⋆ If there is an abstract edge $((q,R),k) \longrightarrow_{\widehat{\mathcal{D}}} \rho$, then $\mathfrak{R}^k_{(q,R)}$ is

$$Pre_{[\![\mathcal{H}]\!]}\Big(\mathcal{S}^k_{(q,R)}, \big\{\mathfrak{R}^{k+1}_{(q',R')}\big\}_{(q',R')\in Q_{k+1}}, \{\rho((q',R'),k+1)\}_{(q',R')\in Q_{k+1}}\Big).$$

Example 7.6. Consider the LDAG $\widehat{\mathcal{D}}$ shown in Figure 3. We compute the concrete set of states $\mathcal{S}^k_{(q,R)}$ and backward reach sets $\mathfrak{R}^k_{(q,R)}$ for each abstract state (q,R) at layer k, from the bottom to the top layer. The continuous part of $\mathcal{S}^k_{(q,R)}$ and $\mathfrak{R}^k_{(q,R)}$ for each abstract state of $\widehat{\mathcal{D}}$ are expressed in the lower and upper part of the circle, respectively, in Figure 4. The concrete sets of states are computed as $\mathcal{S}^k_{(q,R)} = \alpha^{-1}_{\mathcal{R}}(q,R) = (q,R)$ from $\alpha_{\mathcal{R}}$ given in Example 7.1. For example, for the abstract state (q_3,R) at layer 1, $\mathcal{S}^1_{(q_3,R)} = (q_3,[0,4])$ and its continuous part is $[0,4]$ which is written in lower part of its circle. Backward reach sets are computed according to its inductive definition. There are three cases. (a) The abstract state is located at the last layer of $\widehat{\mathcal{D}}$. For example, abstract state (q_1,R) is located at layer 3 which is the depth of $\widehat{\mathcal{D}}$. Hence $\mathfrak{R}^3_{(q_1,R)} = \mathcal{S}^3_{(q_1,R)} = (q_1,[0,4])$. (b) There is no probabilistic edge from the abstract state and it is not located at the last layer. For example, there is no probabilistic edge from (q_4,R) at layer 2. Hence, $\mathfrak{R}^2_{(q_4,R)} = \mathcal{S}^2_{(q_4,R)} = (q_4,[0,4])$. (c) All those abstract states from which there is a probablistic edge. For example, abstract state (q_3,R) at layer 2 has a probabilistic edge to (q_1,R) and (q_4,R) at layer 3 with probabilities 0.5 and 0.5, respectively. The backward reach set $\mathfrak{R}^2_{(q_3,R)}$ is $Pre_{[\![\mathcal{H}]\!]}(\mathcal{S}^2_{(q_3,R)}, \{\mathcal{S}^3_{(q_1,R)}, \mathcal{S}^3_{(q_4,R)}\}, \{0.5, 0.5\})$. From Definition 6.3, $Pre_{[\![\mathcal{H}]\!]}(\mathcal{S}^2_{(q_3,R)}, \{\mathcal{S}^3_{(q_1,R)}, \mathcal{S}^3_{(q_4,R)}\}, \{0.5, 0.5\})$ is computed as follows. First, for the edge from q_3 to q_1, we compute the set G_1 of continuous states which satisfy the guard and transition to (q_1,R). Since the guard constraint is $x \in [3,4]$, $R = [0,4]$ and the reset is identity, we obtain $G_1 = [3,4]$. Similarly, the continuous states which can transition to q_4 is given by $G_2 = [3,4]$. Hence, the continuous states from which the probabilistic edge can be taken is given by the set $G = G_1 \cap G_2 = [3,4]$. The set of all continuous states in $[0,4]$ which can reach G with rate $-1 \le \dot{x} \le 0$ is $[3,4]$, since, all states in $[3,4]$ can reach G by following the rate 0, and no states in $[0,3)$ can reach G because there are no positive rates. Hence, $Pre_{[\![\mathcal{H}]\!]}(\mathcal{S}^2_{(q_3,R)}, \{\mathcal{S}^3_{(q_1,R)}, \mathcal{S}^3_{(q_4,R)}\}, \{0.5, 0.5\})$ is $(q_3,[3,4])$ which is the set of states in $\mathcal{S}^2_{(q_3,R)}$ from which there is a concrete probabilistic edge to $\mathcal{S}^3_{(q_1,R)}$ and $\mathcal{S}^3_{(q_4,R)}$ with probabilities 0.5 each.

The next proposition states that if any of the backward reach sets becomes empty there there is no concrete counterexample corresponding to the abstract counterexample.

Proposition 7.7. *$\alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$ is an empty set if and only if there exists k such that $\mathfrak{R}^k_{(q,R)} = \emptyset$ for some $((q,R),k) \in \mathcal{S}_{\widehat{\mathcal{D}}}$.*

Proof. If $\alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$ is an empty set, then $\mathfrak{R}^k_{(q,R)}$ is empty for $k = 0$. Suppose there exists k such that $\mathfrak{R}^k_{(q,R)} = \emptyset$ for some $((q,R),k) \in \mathcal{S}_{\widehat{\mathcal{D}}}$. This implies that $\mathfrak{R}^{k'}_{(q',R')}$ is also an empty set for all those states $((q',R'),k')$ which are on the path from the root state of $\widehat{\mathcal{D}}$ to the state $((q,R),k)$. In particular, we have $\mathfrak{R}^0_{(q_0,R_0)} = \emptyset$, which by definition implies that $\alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$ is an empty set. □

If $\alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$ is not empty, then there is a valid counterexample $\mathcal{D} \in Exec([\![\mathcal{H}]\!])$ for the concrete system $\mathcal{H}$, that is, $\mathcal{H}$ is not probabilistically safe. However, if $\alpha_{\mathcal{R}}^{-1}(\widehat{\mathcal{D}})$ is empty, then $\widehat{\mathcal{D}}$ is a spurious counterexample. Hence, we use the information from the analysis of the counterexample $\widehat{\mathcal{D}}$ to refine the concrete system. We define the following which will be useful in defining the refinement.

Definition 7.8. A point of refinement denoted as $por_{Abs(\mathcal{H},\mathcal{R})}(\widehat{\mathcal{D}})$, is defined as the largest layer l of $\widehat{\mathcal{D}}$ such that there is a state $((q,R),l) \in \mathcal{S}_{\widehat{\mathcal{D}}}$ where $\mathfrak{R}^l_{(q,R)}$ is an empty set. A probabilistic edge $(((q,R),l'),\rho) \in \longrightarrow_{\widehat{\mathcal{D}}}$ is spurious if $\mathfrak{R}^{l'}_{(q,R)} = \emptyset$ and $l' = l$.

In Figure 4, the largest layer at which the backward reach set is empty in $l = 0$. Hence, the point of refinement is 0 and the spurious edge is $(((q_1,R),0),\rho)$, where $\rho((q_2,R),1) = \rho((q_3,R),1) = 0.5$.

7.4 Refinement

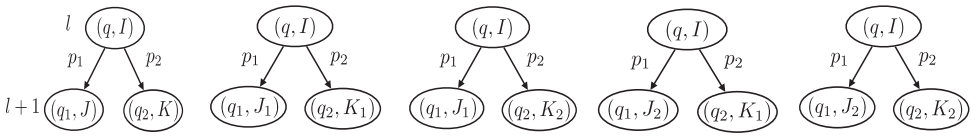
In this section, we provide a method to refine the abstract system $Abs(\mathcal{H},\mathcal{R})$ to eliminate the spurious counterexample $\widehat{\mathcal{D}}$. Let us fix a point of refinement l for $\widehat{\mathcal{D}}$ and a refinement of partition $\mathcal{R}$ denoted as $\mathcal{R}_1$. Consider a function mapping the refinement to the current abstraction, given by $\beta_{(\mathcal{R}_1,\mathcal{R})} : \mathcal{R}_1 \to \mathcal{R}$ where for all $(q,R) \in \mathcal{R}_1$, $(q,R) \subseteq \beta_{(\mathcal{R}_1,\mathcal{R})}(q,R)$. We need to eliminate the spurious counterexample from the abstract system $Abs(\mathcal{H},\mathcal{R}_1)$. Hence, we define a progressive refinement which ensures that the refinement makes progress towards eliminating the counterexample $\widehat{\mathcal{D}}$. It essentially captures the intuition that every abstract counterexample in the refinement that corresponds to $\widehat{\mathcal{D}}$ has a point of refinement which is lower in the LDAG.

Definition 7.9 (Progressive Refinement). A refinement $\mathcal{R}_1$ of $\mathcal{R}$ is a progressive refinement if we have

$$por_{Abs(\mathcal{H},\mathcal{R}_1)}(\mathcal{D}) > por_{Abs(\mathcal{H},\mathcal{R})}(\widehat{D}), \text{ for all } \mathcal{D} \in \beta^{-1}_{(\mathcal{R}_1,\mathcal{R})}(\widehat{\mathcal{D}}).$$

Our objective is to find a progressive refinement $\mathcal{R}_1$ of $\mathcal{R}$. Since the main reason for a probabilistic edge $((q,R),l),\rho)$ to be spurious is the emptiness of the backward reach set $\mathcal{R}^l_{(q,R)}$, our broad approach is to find a splitting of the nodes in the $l+1$ layer that eliminate the spuriousness at layer $l+1$. This is achieved by splitting the nodes in layer l to separate the post of $\mathcal{S}^l_{(q,R)}$ from the backward reach sets of layer $l+1$. Hence, we define a function *Post*, which is along the lines of *Pre*.

Definition 7.10. Given a set S, a tuple $(p_1,p_2,\ldots,p_n)$, we define a function $Post_{[\![\mathcal{H}]\!]}(S,(p_1,\ldots,p_n))$ as $\{(s_1,s_2,\ldots,s_n) \mid \exists\, s \in S,\ (s,\rho) \in \longrightarrow_{[\![\mathcal{H}]\!]}$ such that $\rho(s_i) = p_i$ for all $i \in [n]\}$.



(a) Spurious Edge (b) Edge 1 (c) Edge 2 (d) Edge 3 (e) Edge 4

Fig. 5. Potential edges.

For any spurious edge $(((q,R),l),\rho)$, backward reach set $\mathfrak{R}^{l}_{(q,R)} = \emptyset$, that is, $Pre_{[\![\mathcal{H}]\!]}(\mathcal{S}^{l}_{(q,R)}, \{\mathfrak{R}^{l+1}_{(q',R')}\}_{(q',R')\in\mathcal{Q}_{l+1}}, (p_1,\ldots,p_n)) = \emptyset$, where $(p_1,\ldots,p_n) \in \times_{(q',R')\in\mathcal{Q}_{l+1}}\{\rho((q',R'),l+1)\}$ for some $n \in \mathbb{N}$. From the definition of *Post* function as given in Definition 7.10, $Post_{[\![\mathcal{H}]\!]}(\mathcal{S}^{l}_{(q,R)}, (p_1,\ldots,p_n))$ intersected with the tuple of concrete states corresponding to the abstract states in layer $l+1$ is not empty, because there exists an abstract probabilistic edge corresponding to the spurious edge. Also, we obtain that no common element in the post of the concrete set of states of the abstract state (q,R) at layer l with respect to the spurious edge and Cartesian product of backward reach sets of all target abstract states (q',R') at layer $l+1$ of the spurious edge. This is formally stated in the following proposition.

Proposition 7.11. *Given a probabilistic edge* $(((q,R),l),\rho) \in \longrightarrow_{\widehat{\mathcal{D}}}$ *(spurious) and* $(p_1,\ldots,p_n) \in \times_{(q',R')\in\mathcal{Q}_{l+1}}\{\rho((q',R'),l+1)\}$ *for some* $n \in \mathbb{N}$*, we have*

$$Post_{[\![\mathcal{H}]\!]}(\mathcal{S}^{l}_{(q,R)}, (p_1,\ldots,p_n)) \cap \underset{(q',R')\in\mathcal{Q}_{l+1}}{\times} \mathfrak{R}^{l+1}_{(q',R')} = \emptyset. \tag{1}$$

It might be possible that intersection between the projection of the post for an abstract state and backward reach set for the same abstract state is not empty. This is formally written as a remark.

Remark 1. Given a probabilistic edge $(((q,R),l),\rho) \in \longrightarrow_{\widehat{\mathcal{D}}}$ (spurious), it is not necessary that the following statement is true. Let $(p_1,\ldots,p_n) \in \times_{(q',R')\in\mathcal{Q}_{l+1}}\{\rho((q',R'),l+1)\}$ for some $n \in \mathbb{N}$.

$$Proj(Post_{[\![\mathcal{H}]\!]}(\mathcal{S}^{l}_{(q,R)}, (p_1,\ldots,p_n)), ((q'',R''),l+1)) \cap \mathfrak{R}^{l+1}_{(q'',R'')}$$

need not be empty for all $(q'',R'') \in \mathcal{Q}_{l+1}$.

Example 7.12. We illustrate Remark 1 using a counterexample. Consider the spurious edge E shown in Figure 4. The post of the abstract state $(q_1,[0,4])$ is $\mathcal{B} = (q_2,[0,2]) \times (q_3,[0,2])$. From Figure 4, we have $\mathfrak{R}^{1}_{(q_2,R)} = (q_2,[1,4])$ and $\mathfrak{R}^{1}_{(q_3,R)} = (q_3,[3,4])$. This implies that $\mathcal{B} \cap (\mathfrak{R}^{1}_{(q_2,R)} \times \mathfrak{R}^{1}_{(q_3,R)}) = \emptyset$. Next, the projection of $\mathcal{B}$ for the abstract state (q_2,R) and (q_3,R) at layer 1 are $Proj(\mathcal{B},((q_2,R),1)) = (q_2,[0,2])$ and $Proj(\mathcal{B},((q_3,R),1)) = (q_3,[0,2])$, respectively. Note that although $Proj(\mathcal{B},((q_3,R),1)) \cap \mathfrak{R}^{1}_{(q_3,R)} = \emptyset$, $Proj(\mathcal{B},((q_2,R),1)) \cap \mathfrak{R}^{1}_{(q_2,R)} \neq \emptyset$.

Next, we provide the details about the separation. Let us consider a spurious edge E with the point of refinement l shown in Figure 5(a) where the probability of transitions to abstract states (q_1,J) and (q_2,K) are p_1 and p_2, respectively. Let $\mathcal{S}^{l}_{(q,I)}$, $\mathcal{S}^{l+1}_{(q_1,J)}$ and $\mathcal{S}^{l+1}_{(q_2,K)}$ be concrete set of states for the abstract states (q,I), (q_1,J) and (q_2,K), respectively, and $\mathfrak{R}^{l}_{(q,I)}$, $\mathfrak{R}^{l+1}_{(q_1,J)}$ and $\mathfrak{R}^{l+1}_{(q_2,K)}$ be the backward reach set for the abstract states (q,I), (q_1,J) and (q_2,K), respectively. Let X be the post of $\mathcal{S}^{l}_{(q,I)}$, and $Y = \mathfrak{R}^{l+1}_{(q_1,J)} \times \mathfrak{R}^{l+1}_{(q_2,K)}$. Let X_1 and X_2 be the projection of X for the abstract states (q_1,J) and (q_2,K), respectively, and Y_1 and Y_2 be the projection of Y for the abstract states (q_1,J) and (q_2,K) at layer $l+1$, respectively, that is, $Y_1 = \mathfrak{R}^{l+1}_{(q_1,J)}$ and $Y_2 = \mathfrak{R}^{l+1}_{(q_2,K)}$.

Case (a). When $X_i \cap Y_i = \emptyset$ for $i = 1,2$, X_1, Y_1 and X_2, Y_2 need to be separated by the partition of $\mathcal{S}^{l+1}_{(q_1,J)}$ and $\mathcal{S}^{l+1}_{(q_2,J)}$, respectively. Next, we show that such partition eliminates the spurious

edge. Let $\mathcal{S}^{l+1}_{(q_1,J)}$ and $\mathcal{S}^{l+1}_{(q_2,K)}$ be partitioned into $\mathcal{S}^{l+1}_{(q_1,J_1)}$, $\mathcal{S}^{l+1}_{(q_1,J_2)}$ and $\mathcal{S}^{l+1}_{(q_2,K_1)}$, $\mathcal{S}^{l+1}_{(q_2,K_2)}$, respectively, such that $X_1 \subseteq \mathcal{S}^{l+1}_{(q_1,J_1)}$, $Y_1 \subseteq \mathcal{S}^{l+1}_{(q_1,J_2)}$ and $X_2 \subseteq \mathcal{S}^{l+1}_{(q_2,K_1)}$ and $Y_2 \subseteq \mathcal{S}^{l+1}_{(q_2,K_2)}$. There are four potential probabilistic edges corresponds to the spurious edge shown in Figures 5(b), 5(c), 5(d) and 5(e).

(1) Edge shown in Figure 5(b) exists in the refined system because $X \cap (\mathcal{S}^{l+1}_{(q_1,J_1)} \times \mathcal{S}^{l+1}_{(q_2,K_1)}) \neq \emptyset$. However, the point of refinement must be $l+1$ because backward reach set $\mathfrak{R}^{l+1}_{(q_1,J_1)} = \emptyset$ because $\mathfrak{R}^{l+1}_{(q_1,J_1)} \subseteq \mathcal{S}^{l+1}_{(q_1,J_1)}$, $\mathfrak{R}^{l+1}_{(q_1,J_1)} \subseteq Y_1$ and $\mathcal{S}^{l+1}_{(q_1,J_1)} \cap Y_1 = \emptyset$.
(2) Edge shown in Figure 5(c) does not exist in the refined system because $X_2 \cap \mathcal{S}^{l+1}_{(q_2,K_2)} = \emptyset$.
(3) Edge shown in Figure 5(d) does not exist in the refined system because $X_1 \cap \mathcal{S}^{l+1}_{(q_1,J_2)} = \emptyset$.
(4) Edge shown in Figure 5(e) does not exist in the refined system because $X_1 \cap \mathcal{S}^{l+1}_{(q_1,J_1)} = \emptyset$ and $X_2 \cap \mathcal{S}^{l+1}_{(q_2,K_2)} = \emptyset$.

Hence, such separation guarantees that the spurious edge gets eliminated.

Case (b). When either $X_1 \cap Y_1 \neq \emptyset$ or $X_2 \cap Y_2 \neq \emptyset$ is true, $X_1 \setminus Y_1$, Y_1 and $X_2 \setminus Y_2$, Y_2 need to be separated by the partition of $\mathcal{S}^{l+1}_{(q_1,J)}$ and $\mathcal{S}^{l+1}_{(q_2,J)}$, respectively. Next, we show that spurious edge gets eliminated by such partition. Assume that $X_1 \cap Y_1 \neq \emptyset$. Note that for a spurious edge, there must be at least one target abstract state such that intersection between the projection of post with respect to the target abstract state and its backward reach set is empty. This implies that $X_2 \cap Y_2 = \emptyset$.

(a) Edge shown in Figure 5(b) exists in the refined system, but the point of refinement must be $l+1$ due to the same reason given in case (a)(1).
(b) Edges shown in Figures 5(c), 5(e) do not exist in the refined system due to the same reason given for case (a)(2).
(c) Edge shown in Figure 5(d) exist in the refined system but the point of refinement must be $l+1$ due to the same reason given in case (a)(1).

From case (a) and case (b), we have identified that $X_i \setminus Y_i$ and Y_i, $i = 1, 2$ needs to be separated. Next, we provide strategies for partitioning the concrete states associated with the abstract states such that there identified sets must be separated by the partition.

Definition 7.13. Let $(((q,R),l),\rho)$ be a probabilistic edge (spurious) and Q_{l+1} be a set of all abstract states which are at layer $l+1$. A refinement $\mathcal{R}_1$ eliminates a spurious edge $(((q,R),l),\rho)$ if for each $(q',R') \in Q_{l+1}$, there is no partition element $(q',R_1) \in \mathcal{R}_1$ such that the following two conditions hold: let $(p_1,\ldots,p_n) \in \times_{(q'',R'')\in Q_{l+1}}\{\rho((q'',R''),l+1)\}$ for some $n \in \mathbb{N}$.

C1: $R_1 \cap (Proj(Post_{[\![\mathcal{H}]\!]}(\mathcal{S}^l_{(q,R)},(p_1,\ldots,p_n)),((q',R'),l+1)) \setminus \mathfrak{R}^{l+1}_{(q',R')}) \neq \emptyset$;
C2: $R_1 \cap \mathfrak{R}^{l+1}_{(q',R')} \neq \emptyset$.

The strategies in Definition 7.13 are basically restrictions on the partition of the concrete set of states corresponding to the abstract states which make sure that the identified sets are not shared by the same partition element. Since the spurious edge gets eliminated by such a partitioning, the progress of the refinement is guaranteed. We formally state it in the following theorem.

Theorem 7.14. *Given a refinement $\mathcal{R}_1$ of $\mathcal{R}$, if $\mathcal{R}_1$ eliminates the spurious edge, then $\mathcal{R}_1$ is a progressive refinement.*

Any such splitting as given by Definition 7.13 will eliminate the counterexample. Note that Definition 7.13 requires that there is no partition element which for any location q intersects with both the projection of *Post* to q and the reach set $\mathfrak{R}$ corresponding to q. Hence, for every partition

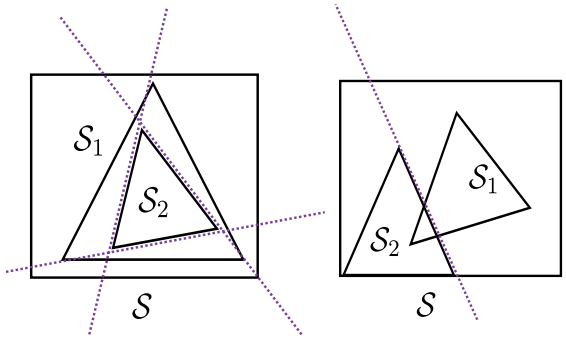


Fig. 6. Illustration of Algorithm 2.

element which intersects with both, we need a strategy to separate the two sets, say, S_1 and S_2. One possible way to have such splitting can be obtained by Algorithm 2. Algorithm 2 aims to partition a polyhedral set $\mathcal{S}$ for given two polyhedral sets $\mathcal{S}_1 \subseteq \mathcal{S}$ and $\mathcal{S}_2 \subseteq \mathcal{S}$ such that $\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2$ does not share the same partition element of the partition of $\mathcal{S}$.

ALGORITHM 2: Refine a partition $\mathcal{P}$ to separate$\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2$

Input: $\mathcal{P}$ - a partition, $\mathcal{S}_1 \subseteq \mathcal{S}$ - a polyhedral set, $\mathcal{S}_2 \subseteq \mathcal{S}$ - a polyhedral set

Output: $\mathcal{P}$ - a partition of $\mathcal{S}$ that separates $\mathcal{S}_1 \setminus \mathcal{S}_2$ and $\mathcal{S}_2$

```
1 begin
2     Let S'_1 = S_1 \ S_2
3     while ∃ P ∈ P such that P ∩ S_1 ≠ ∅ and P ∩ S'_1 ≠ ∅ do
4         if ∃ c ∈ Cons(S_2) such that [[c]]‾ ∩ S'_1 ≠ ∅ then
5             P = P \ {P}
6             P = P ∪ {P ∩ [[c]]}
7             P = P ∪ {P ∩ [[c]]‾}
8     return P
9 end
```

Algorithm 2 checks whether there is a partition element P in the current partition $\mathcal{P}$ which overlaps with both $\mathcal{S}_1$ and $\mathcal{S}_1 \setminus \mathcal{S}_2$ at line 3. If so, then we check if there exists a constraint $c \in Cons(\mathcal{S}_2)$ such that $\mathcal{S}_1 \setminus \mathcal{S}_2$ overlaps with the complement of the set $[\![c]\!]$, then we split P into two polyhedral sets $P \cap [\![c]\!]$ and $P \cap \overline{[\![c]\!]}$ and add them into $\mathcal{P}$ at lines 5–7. We repeat 4–7 until we come up $\mathcal{P}$ such that no partition element of $\mathcal{P}$ overlaps with both $\mathcal{S}_1$ and $\mathcal{S}_1 \setminus \mathcal{S}_2$. We have illustrated Algorithm 2 in Figure 6. In the left picture of Figure 6, all the constraints of $\mathcal{S}_2$ are required to separate $\mathcal{S}_2$ and $\mathcal{S}_1 \setminus \mathcal{S}_2$ and it results in 7 partition elements of $\mathcal{S}$. However, in the right picture of Figure 6, one constraint is enough to separate $\mathcal{S}_2$ and $\mathcal{S}_1 \setminus \mathcal{S}_2$, and it results in two partition elements.

Example 7.15. We illustrate the separation based on the spurious edge E shown in Figure 4. From Examples 7.6, 7.12, we have $\mathcal{S}^1_{(q_2,R)} = (q_2, [0, 4])$, $\mathcal{S}^1_{(q_3,R)} = (q_3, [0, 4])$, $\Re^1_{(q_2,R)} = (q_2, [1, 4])$, $\Re^1_{(q_3,R)} = (q_2, [3, 4])$, and the post of the abstract state $(q_1, [0, 4])$ is $\mathcal{B} = (q_2, [0, 2]) \times (q_3, [0, 2])$. Let $\mathcal{B}_{(q_2,R)}$, $\mathcal{B}_{(q_3,R)}$ be the projection of $\mathcal{B}$ for the abstract states (q_2, R), (q_3, R), respectively, at layer 1. Then, we use Algorithm 2 to partition $\mathcal{S}_{(q_2,R)}$ and $\mathcal{S}_{(q_3,R)}$. We obtain $\{(q_2, R^2_1), (q_2, R^2_2)\}$

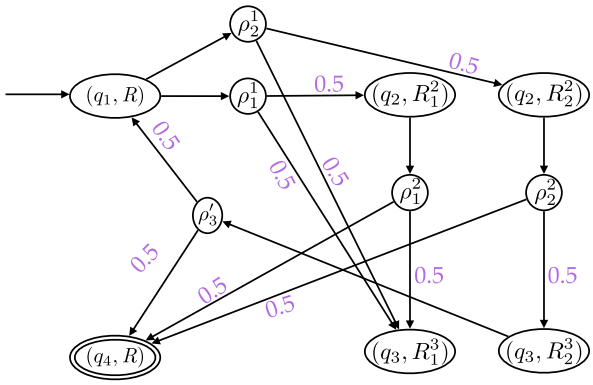


Fig. 7. Markov decision process after refinement.

as a partition of $\mathcal{S}_{(q_2,R)}$, where $R_1^2 = [0,1]$, $R_2^2 = [1,4]$. Similarly, we get $\{(q_3, R_1^3), (q_3, R_2^3)\}$ as a partition of $\mathcal{S}_{(q_3,R)}$, where $R_1^3 = [0,3)$, $R_2^3 = [3,4]$. Thus, we obtain a refinement of the partition $\mathcal{R}$, which is $\mathcal{R}_1 = \{(q_1,R), (q_2,R_1^2), (q_2,R_2^2), (q_3,R_1^3), (q_3,R_2^3), (q_4,R)\}$. Next, we use $\mathcal{R}_1$ for refining the abstraction and we obtain a Markov Decision Process as shown in Figure 7. The probability of reaching final state (q_4, R) from initial state (q_1, R) is less than 0.50. Hence, polyhedral PHS shown in Figure 1 is probabilistically safe with respect to the initial location q_0, a set of final locations $\mathbb{F} = \{q_4\}$, and probability bound $\theta = 0.50$.

8 COMPUTABILITY

In this section, we summarize the CEGAR algorithm and discuss computability and complexity issues.

8.1 CEGAR Algorithm

Algorithm 3 presents the CEGAR framework to check whether a given polyhedral PHS $\mathcal{H}$ is the maximum probabilistically safe with respect to an initial location q_0, a set of final locations $\mathbb{F}$, and a probability bound θ. The function Abstraction takes as input the system $\mathcal{H}$, partition $\mathcal{R}$, the initial state q_0 and the set of final locations $\mathbb{F}$ and returns the MDP abstraction $\mathcal{M} = Abs(\mathcal{H}, \mathcal{R})$ along with the set of abstract states S corresponding to the concrete states $\{q_0\} \times Inv(q_0)$ and the set of abstract states $\mathbb{F}'$ corresponding to the concrete states $\mathbb{F} \times \mathcal{X}$. The function Model-Checking takes the abstract MDP $\mathcal{M}$, abstract initial states S and final state $\mathbb{F}'$ as input and either returns a $Status = \top$, if the abstract system satisfies the property and $\bot$ otherwise. In the latter case, it also returns a counterexample $\widehat{\mathcal{T}}$ in the form of a DTMC, which is then passed on to the function Get_LDAG to extract the LDAG $\widehat{\mathcal{D}}$. The function Validation function validates the counterexample $\widehat{\mathcal{D}}$ using the backward reachability algorithm and checking if $\Re^k_{(q,R)} = \emptyset$ for some $((q,R),k) \in \mathcal{S}_{\widehat{\mathcal{D}}}$. $valid = \top$ if the above condition is satisfied in which case a concrete counterexample exists, otherwise, $\widehat{\mathcal{D}}$ is a spurious abstract counterexample, in which case some spurious information SI is returned. It first computes the spurious edge $SE = (((q,R),l),\rho) \in \longrightarrow_{\widehat{\mathcal{D}}}$ where l is the largest number satisfying $\Re^l_{(q,R)} = \emptyset$ for some (q,R) and ρ. Then, SI consists of three sets S_1, S_2 and S_3, where $S_1 = \times_{((q',R'),l+1) \in support(\rho)} \mathcal{S}^{l+1}_{(q',R')}$, $S_2 = Post(S^l_{(q,R)}, p_1, p_2, \ldots, p_n)$, where $p_i = \rho((q_i,R_i), l+1)$, and $S_3 = \times_{((q',R'),l+1) \in support(\rho)} \Re^{l+1}_{(q',R')}$. Refinement($\mathcal{R}$, SI) runs Algorithm 2 for partitioning $Proj(S_1, ((q,R), l+1))$ for each abstract state $((q,R), l+1) \in support(\rho)$

ALGORITHM 3: CEGAR Algorithm

Input: $\mathcal{H}$ - polyhedral PHS, q_0 - initial location, $\mathbb{F}$ - a set of final locations, θ - probability bound, $\mathcal{R}$ - partition of the state space

Output: $\top/\bot$ based on satisfaction/rejection of the specification with respect to q_0, $\mathbb{F}$, θ

```
1  begin
2      Sat = None
3      while Sat = None do
4          M, S, F' = Abstraction(H, R, q0, F)
5          T̂, Status = Modelchecking(M, s, F', θ)
6          if Status = ⊤ then
7              Sat = ⊤
8              break
9          else
10             D̂ = Get_LDAG(T̂, F',S, θ)
11             SI, valid = Validation(H, D̂, R)
12             if valid = ⊤ then
13                 Sat = ⊥
14                 break
15             else
16                 R1 = Refinement(R, SI)
17                 R = R1
18 end
```

with respect to $Proj(S_2, ((q, R), l + 1), Proj(S_3, ((q, R), l + 1)$. Then, for each for each abstract state $((q, R), l + 1) \in support(\rho)$, the partition element $Proj(S_1, ((q, R), l + 1)$ of $\mathcal{R}$ is replaced by all its partition elements obtained from Algorithm 2. The CEGAR loop is repeated until the specification is refuted or satisfied, in general, it is not guaranteed to terminate. Hence, it is a semi-decision procedure.

8.2 Complexity Analysis

In this section, we discuss the computability and complexity of the different steps in the CEGAR algorithm. The crux of Algorithm 3 is the computation of *Pre* and *Post* functions, corresponding to computing the one step backward and forward reach sets. When the dynamics is polyhedral inclusion $\dot{x} \in P$ for a polyhedral set P, the *Pre* and *Post* with respect to a polyhedral set of states S, results in a polyhedral set. This can be computed by writing a finite set of linear constraints that the states before and after a transition need to satisfy, and eliminating one of them (depending on whether we are computing *Pre* or *Post*). This corresponds to computing intersections, checking satisfiability and variable elimination in a set of linear constraints. The first two operations can be performed efficiently (polynomial in the size of the constraints), while variable elimination is more expensive and can be performed using Fourier–Motzkin elimination whose complexity is $O(4(\frac{n_c}{4})^{2^{n_v}})$, where n_c is the number of linear constraints in a polyhedral set and n_v is the number of variables that need to be eliminated. It also leads to a exponential blow-up in the number of constraints with respect to the number of eliminated variables. In the rest of the section, we present the complexity analysis assuming a cost of C for each *Pre* and *Post* computation. We realize that

C depends on the size of the representation of the intermediate reach sets which can themselves grow exponentially as the algorithm proceeds, however, we would like to still characterize the behavior of the rest of the algorithm.

We note that for more general dynamics, the reach set is not a polyhedral set and could require exponential functions for representation, for instance, for linear dynamics such as $\dot{x} = Ax$ or other non-linear dynamics. Hence, though the CEGAR framework in Algorithm 3 can be applied, one will need to resort to over-approximate computations of *Pre* and *Post* for abstraction construction and deducing spuriousness of the counterexample.

Next, we discuss the complexity of each step presented in Algorithm 3. Let us consider an $n-$dimensional polyhedral PHS $\mathcal{H}$, and a partition $\mathcal{R}$ of its state space $Q \times X$. Let $|\mathcal{R}| = k_1 \times k_2$ where $k_1 = |Q|$ and k_2 is the number of distinct partition elements of $\mathcal{X}$.

In the abstraction, each abstract edge has a source of the form (q, R) and a set of target states of the form $(q_1, R_1), \ldots, (q_{k_1}, R_{k_1})$. We need to check if each such potential abstract edge corresponds to a concrete edge, which can be computed by one *Pre* operation followed by checking emptiness. Hence, the cost of abstraction is given by $O(Ck_2^{k_1+1})$, and size (representation) of the output MDP $\mathcal{M}$ is $O(k_2^{k_1+1})$. The complexity of Model-Checking function is polynomial in the size of the MDP $\mathcal{M}$. The GET_LDAG algorithm runs until enough paths are found. There is a priori no bound on the length of paths explored, however, it is guaranteed to terminate and the time complexity is polynomial in d, which is the length of the unrolling. Then the complexity of validation and computation of SI is $O(DC)$, since the validation requires one *Pre* computation and one emptiness checking operation (which is cheaper than *Pre* computation) corresponding to every node in $\widehat{\mathcal{D}}$. The Refinement function calls the partitioning algorithm at most once for every location. The complexity of the partitioning algorithms is $O(n_c n_p)$, where n_c number of constraints in the representation of $\Re$ sets, and n_p is the number of partition elements. However, the number of elements of the refined partition could be exponential in n_c.

9 EXPERIMENTS

In this section, we provide the details of the implementation and experimental analysis. We have implemented the CEGAR framework for polyhedral PHS in a Python toolbox called Procegar. We use Parma Polyhedra Library (PPL) [8] to compute the abstract MDP, and PRISM [30] model checker to verify the probabilistic safety specification on the abstract system. PRISM generates a counterexample in the form of a DTMC. We unroll the DTMC and convert it into an LDAG which we validate with respect to the polyhedral PHS using PPL [8]. The refinement module generates a new partition of the state space that eliminates the counterexample, and it is implemented using PPL [8]. We compare our experimental evaluation of probabilistic safety analysis with tool ProHVer [45], which has been performed on Ubuntu 12.04 OS, Intel R©Pentium(R) CPU B960 with 2.20GHz× 2 Processor and 2GB RAM.

We consider two examples, one corresponding to a grid world, and the other an oscillator-filter. For the grid world example, we construct grids with $n \times n$ cells, where we have polyhedral dynamics $\dot{x} - \dot{y} <= 0$, $2\dot{x} - \dot{y} >= 0$ in each cell. A probabilistic edge is defined from a cell to two of its adjacent cells with equal probability. here, we consider the specification, where we choose cell $(1, 1)$ as an initial location with continuous states $[0, 0.5] \times [0, 0.5]$, and cell $(2, 2)$ as a final location with continuous states $[1.5, 2] \times [1, 1.5]$. Next, for the 2-dimensional oscillator and m-filter benchmark, we modify the deterministic version of the oscillator-filter benchmark [18] by introducing probabilities and additional reset constraints. Note that the deterministic version of the benchmark is a linear PHS. So, we first convert the benchmark into polyhedral PHS by transforming the linear dynamics into polyhedral dynamics by hybridization over the

Table 1. Verification Results for the Grid World ($n = 2, K = 2$)

			Procegar						ProHVer		
Rows	*Grid Size*	θ	*Size(init)*	*Size(final)*	*Status*	T_{A+CE+V}(*Sec.*)	T_{ver}(*Sec.*)	P_T(*Sec.*)	*Pr*	*Status*	P_T(*Sec.*)
1	2×2	0.25	(4,3)	(9,6)	$\bot$	6.56	3.13	9.69	1	*U*	0.03
2		0.50	(4,3)	(9,6)	$\top$	3.31	3.08	6.39	1	*U*	0.03
3	4×4	0.10	(16,15)	(23,19)	$\bot$	6.20	3.18	9.38	0.25	*U*	0.057
4		0.25	(16,15)	(23,19)	$\top$	3.17	3.08	6.25	0.25	$\top$	0.057
5	6×6	0.10	(36,35)	(43,39)	$\bot$	6.32	3.07	9.39	0.25	*U*	0.085
6		0.25	(36,35)	(43,39)	$\top$	3.20	2.96	6.16	0.25	$\top$	0.085
7	8×8	0.10	(64,63)	(71,67)	$\bot$	6.30	3.17	9.47	0.25	*U*	0.122
8		0.25	(64,63)	(71,67)	$\top$	3.26	2.94	6.20	0.25	$\top$	0.122

Table 2. Verification Results for the Oscillator-filter ($K = 1$)

			Procegar						ProHVer		
m	*dim*	θ	*Size(init)*	*Size(final)*	*Status*	T_{A+CE+V}(*Sec.*)	T_{ver}(*Sec.*)	P_T(*Sec.*)	*Pr*	*Status*	P_T(*Sec.*)
1	3	0.10	(4,3)	(4,3)	$\bot$	3.25	1.51	4.76	0.73	*U*	11.66
		0.20	(4,3)	(4,3)	$\top$	0.01	1.45	1.46	0.73	*U*	11.66
2	4	0.10	(4,3)	(4,3)	$\bot$	4.70	1.46	6.16	0.73	*U*	711.96
		0.20	(4,3)	(4,3)	$\top$	0.01	1.48	1.49	0.73	*U*	711.96
3	5	0.10	(4,3)	(4,3)	$\bot$	36.11	1.65	37.76	0.73	*U*	4385.63
		0.20	(4,3)	(4,3)	$\top$	0.01	1.55	1.56	0.73	*U*	4385.63
4	6	0.10	(4,3)	(4,3)	$\bot$	1273.30	1.55	1274.85	—	—	*TO*
		0.20	(4,3)	(4,3)	$\top$	0.01	1.88	1.89	—	—	*TO*

invariant set. In addition, we consider the state space for each variable as the interval $[-3, 3]$. For the specification, we consider $(l_1, \{-0.5 \le x \le 0,\ 0 \le y \le 0.35\})$ as the initial set of states and $(l_4, \{0 \le x \le 0.5, -0.35 \le y \le 0\})$ as the final set of states. For all the experiments, we are interested in verifying $\max_{s\in\mathbb{I}} Prob_{\max}([\![\mathcal{H}]\!], s, \mathbb{F}) \le \theta$ for different values of probability bound θ, where $\mathbb{I}$ is the set of initial states and $\mathbb{F}$ is the set of final states.

We perform experiments using tools Procegar and ProHVer [45] and compare the experimental results. For the probabilistic safety analysis using tool ProHVer [45], we first compute the maximum probability of reaching a set of final states from a set of initial states and check whether the maximum probability of reachability is less than or equal to θ.

In Table 1 and Table 2, the results of analysis of the grid world and the oscillator-filter are summarized, respectively. In the tables, θ shows the probability bound; *Size(init)*, *Size(final)* show the size of initial and final abstract system which are expressed as a pair $(|V|, |E|)$, where $|V|$, $|E|$ are number of vertices and edges, respectively. *Status* represents whether a given specification is satisfied ($\top$), non-satisfied ($\bot$) or the verification result is inconclusive (U). T_{A+CE+V} shows the total time (among all iterations) taken for the abstraction, counterexample simplification, and validation steps; T_{ver} represents the total time (among all iterations) for checking the property by PRISM and P_T represents the total time taken for the safety verification. All times are measured in seconds. *Pr* represents the value of the maximum probability of reachability for a given specification in ProHVer [45]. *dim*, K represent dimension of the system and number of iterations in CEGAR, respectively. m shows the number of filters for 2-dimensional oscillator and m-filter benchmark.

We observe that in both Table 1 and Table 2, Procegar concludes safety of the specification for different values of θ for all grid sizes, however, in Table 1 ProHVer [45] is unable to conclude safety

in Rows 1, 2, 3, 5, 7, and in Table 2, ProHVer [45] is unable to conclude safety of the specification for $m = 1, 2, 3$. Also, ProHVer [45] did not terminate for $m = 4$ within a timeout of 75 minutes. Procegar is able to find proofs in many more instances and with less time as compared to ProHVer which does not use any clever refinement techniques. There is no extensive set of benchmarks for probabilistic hybrid systems so we have modified some of the existing benchmarks in the non-probabilistic setting.

10 CONCLUSIONS

In this paper, we have developed a CEGAR based method for probabilistic safety analysis of polyhedral probabilistic hybrid systems. We have implemented the method in a Python toolbox and compared our method with the tool ProHVer [45]. Our experimental analysis demonstrates the advantages of our technique in terms of both being able to verify many more systems as well as being able to validate counterexamples. However, there are several strategies for splitting to achieve progressive refinements, and we will explore some of those in the future work. Also, we will extend our CEGAR algorithm for complex stochastic hybrid systems, where we consider stochastic differential equations that capture stochasticity in the continuous behaviors.

ACKNOWLEDGMENTS

Pavithra Prabhakar was partially supported by NSF CAREER Award No. 1552668 and ONR YIP Award No. N000141712577.

REFERENCES

[1] Alessandro Abate, Alessandro D'Innocenzo, and Maria D Di Benedetto. 2011. Approximate abstractions of stochastic hybrid systems. *IEEE Trans. Automat. Control.*

[2] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. 2008. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica.*

[3] Husain Aljazzar, Holger Hermanns, and Stefan Leue. 2005. Counterexamples for timed probabilistic reachability. In *International Conference on Formal Modeling and Analysis of Timed Systems.*

[4] Husain Aljazzar, Matthias Kuntz, Florian Leitner-Fischer, and Stefan Leue. 2010. Directed and heuristic counterexample generation for probabilistic model checking: a comparative evaluation. In *ICSE Workshop on Quantitative Stochastic Models in the Verification and Design of Software Systems.*

[5] Rajeev Alur, Thao Dang, and Franjo Ivančić. 2003. Counter-example guided predicate abstraction of hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems.*

[6] Rajeev Alur, Thao Dang, and Franjo Ivančić. 2003. Progress on reachability analysis of hybrid systems using predicate abstraction. In *International Workshop on Hybrid Systems: Computation and Control.*

[7] Rajeev Alur, Thao Dang, and Franjo Ivančić. 2006. Counterexample-guided predicate abstraction of hybrid systems. *Theoretical Computer Science.*

[8] Roberto Bagnara, Patricia M. Hill, and Enea Zaffanella. 2008. The parma polyhedra library: Toward a complete set of numerical abstractions for the analysis and verification of hardware and software systems. *Sci. Comput. Program.*

[9] Christos G. Cassandras and John Lygeros. 2006. *Stochastic hybrid systems.* CRC.

[10] Rohit Chadha and Mahesh Viswanathan. 2010. A counterexample-guided abstraction-refinement framework for Markov decision processes. *ACM Transactions on Computational Logic (TOCL).*

[11] Xin Chen, Erika Ábrahám, and Sriram Sankaranarayanan. 2013. Flow*: An analyzer for non-linear hybrid systems. In *International Conference on Computer Aided Verification.*

[12] Edmund Clarke, Ansgar Fehnker, Zhi Han, Bruce Krogh, Joël Ouaknine, Olaf Stursberg, and Michael Theobald. 2003. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *International Journal of Foundations of Computer Science.*

[13] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. 2000. Counterexample-guided abstraction refinement. In *International Conference on Computer Aided Verification.*

[14] Pedro R. D'Argenio, Bertrand Jeannet, Henrik Ejersbo Jensen, and Kim Guldstrand Larsen. 2001. Reachability analysis of probabilistic systems by successive refinements. In *Process Algebra and Probabilistic Methods, Performance Modeling and Verification: Joint International Workshop, PAPM-PROBMIV.*

[15] Martin Fränzle, Ernst Moritz Hahn, Holger Hermanns, Nicolás Wolovick, and Lijun Zhang. 2011. Measurability and safety verification for stochastic hybrid systems. In *International Conference on Hybrid Systems: Computation and Control*.
[16] Martin Fränzle, Holger Hermanns, and Tino Teige. 2008. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In *International Workshop on Hybrid Systems: Computation and Control*.
[17] Goran Frehse. 2005. PHAVer: Algorithmic verification of hybrid systems past HyTech. In *International Workshop on Hybrid Systems: Computation and Control*.
[18] Goran Frehse, Colas Le Guernic, Alexandre Donzé, Scott Cotton, Rajarshi Ray, Olivier Lebeltel, Rodolfo Ripado, Antoine Girard, Thao Dang, and Oded Maler. 2011. SpaceEx: Scalable verification of hybrid systems. In *International Conference on Computer Aided Verification*.
[19] Sicun Gao, Soonho Kong, and Edmund M. Clarke. 2013. dReal: An SMT solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*. Springer.
[20] Tingting Han, Joost-Pieter Katoen, and Damman Berteun. 2009. Counterexample generation in probabilistic model checking. *IEEE Transactions on Software Engineering*.
[21] Tingting Han, Joost-Pieter Katoen, and Alexandru Mereacre. 2008. Approximate parameter synthesis for probabilistic time-bounded reachability. In *Real-Time Systems Symposium*.
[22] Hans Hansson and Bengt Jonsson. 1994. A logic for reasoning about time and reliability. *Formal Aspects of Computing*.
[23] Thomas A. Henzinger, Peter W. Kopke, Anuj Puri, and Pravin Varaiya. 1998. What's decidable about hybrid automata? *Journal of Computer and System Sciences*.
[24] Holger Hermanns, Björn Wachter, and Lijun Zhang. 2008. Probabilistic cegar. In *International Conference on Computer Aided Verification*.
[25] Jianghai Hu, John Lygeros, and Shankar Sastry. 2000. Towards a theory of stochastic hybrid systems. In *International Conference on Hybrid Systems: Computation and Control*.
[26] Sumit K. Jha, Bruce H. Krogh, James E. Weimer, and Edmund M. Clarke. 2007. Reachability for linear hybrid automata using iterative relaxation abstraction. In *International Workshop on Hybrid Systems: Computation and Control*. Springer.
[27] Joost-Pieter Katoen, Daniel Klink, Martin Leucker, and Verena Wolf. 2012. Three-valued abstraction for probabilistic systems. *The Journal of Logic and Algebraic Programming*.
[28] Soonho Kong, Sicun Gao, Wei Chen, and Edmund Clarke. 2015. dReach: δ-reachability analysis for hybrid systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer.
[29] Marta Kwiatkowska. 2003. Model checking for probability and time: From theory to practice. In *Logic in Computer Science, 2003. IEEE Symposium on*.
[30] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification*.
[31] Ratan Lal and Pavithra Prabhakar. 2018. Bounded verification of reachability of probabilistic hybrid systems. In *Quantitative Evaluation of Systems QEST*.
[32] Ratan Lal and Pavithra Prabhakar. 2018. Hierarchical abstractions for reachability analysis of probabilistic hybrid systems. In *Annual Allerton Conference on Communication, Control, and Computing (Allerton)*.
[33] John Lygeros and Maria Prandini. 2010. Stochastic hybrid systems: A powerful framework for complex, large scale applications. *Eur. J. Control*.
[34] Anuj Puri, Vivek S. Borkar, and Pravin Varaiya. 1995. Epsilon-approximation of differential inclusions. In *Hybrid Systems III: Verification and Control, DIMACS/SYCON Workshop on Verification and Control of Hybrid Systems*.
[35] Nima Roohi, Pavithra Prabhakar, and Mahesh Viswanathan. 2016. Hybridization based CEGAR for hybrid automata with affine dynamics. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.
[36] Nima Roohi, Pavithra Prabhakar, and Mahesh Viswanathan. 2017. HARE: A hybrid abstraction refinement engine for verifying non-linear hybrid automata. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*.
[37] J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. 2004. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*, P. Panangaden and F. van Breugel (eds.). American Mathematical Society.
[38] Roberto Segala and Nancy A. Lynch. 1994. Probabilistic simulations for probabilistic processes. In *International Conference on Concurrency Theory*.
[39] Fedor Shmarov and Paolo Zuliani. 2015. Probreach: Verified probabilistic delta-reachability for stochastic hybrid systems. In *International Conference on Hybrid Systems: Computation and Control*.
[40] Jeremy Sproston. 2000. Decidable model checking of probabilistic hybrid automata. In *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*.
[41] Tino Teige and Martin Fränzle. 2009. Constraint-based analysis of probabilistic hybrid systems. *IFAC Proceedings Volumes*.

[42] Ashish Tiwari. 2008. Abstractions for hybrid systems. *FMSD*.
[43] Qinsi Wang, Paolo Zuliani, Soonho Kong, Sicun Gao, and Edmund M. Clarke. 2015. Sreach: A probabilistic bounded delta-reachability analyzer for stochastic hybrid systems. In *Conference on Computational Methods in Systems Biology*.
[44] Majid Zamani, Peyman Mohajerin Esfahani, Rupak Majumdar, Alessandro Abate, and John Lygeros. 2014. Symbolic control of stochastic systems via approximately bisimilar finite abstractions. *IEEE Trans. Automat. Control*.
[45] Lijun Zhang, Zhikun She, Stefan Ratschan, Holger Hermanns, and Ernst Moritz Hahn. 2010. Safety verification for probabilistic hybrid systems. In *International Conference on Computer Aided Verification*.

Received April 2019; revised June 2019; accepted July 2019