Safety Verification for Probabilistic Hybrid Systems

Lijun Zhang[1], Zhikun She[2], Stefan Ratschan[3], Holger Hermanns[4], and Ernst Moritz Hahn[4]

[1] DTU Informatics, Technical University of Denmark, Denmark
[2] LMIB and School of Mathematics and Systems Science, Beihang University, China
[3] Institute of Computer Science, Czech Academy of Sciences, Czech Republic
[4] Department of Computer Science, Saarland University, Germany

Abstract. The interplay of random phenomena and continuous real-time control deserves increased attention for instance in wireless sensing and control applications. Safety verification for such systems thus needs to consider probabilistic variations of systems with hybrid dynamics. In safety verification of classical hybrid systems we are interested in whether a certain set of unsafe system states can be reached from a set of initial states. In the probabilistic setting, we may ask instead whether the probability of reaching unsafe states is below some given threshold. In this paper, we consider probabilistic hybrid systems and develop a general abstraction technique for verifying probabilistic safety problems. This gives rise to the first mechanisable technique that can, in practice, formally verify safety properties of non-trivial continuous-time stochastic hybrid systems—without resorting to point-wise discretisation. Moreover, being based on arbitrary abstractions computed by tools for the analysis of non-probabilistic hybrid systems, improvements in effectivity of such tools directly carry over to improvements in effectivity of the technique we describe. We demonstrate the applicability of our approach on a number of case studies, tackled using a prototypical implementation.

1 Introduction

Conventional hybrid system formalisms [1–4] capture many characteristics of real systems (telecommunication networks, air traffic management, etc.). However, in some modern application areas, the lack of randomness hampers faithful modelling and verification. This is especially true for wireless sensing and control applications, where message loss probabilities and other random effects (node placement, node failure, battery drain) turn the overall control problem into a problem that can only be managed with a certain, hopefully sufficiently large, probability.

The idea of integrating probabilities into hybrid systems is not new, and different models have been proposed, each from its own perspective [5–9]. The most important difference lies in the place where to introduce randomness. One option is to replace deterministic jumps by probability distributions over deterministic jumps. Another option is to replace differential equations in each mode by stochastic differential equations. More general models can be obtained by blending the above two choices, and by combining with memoryless timed probabilistic jumps [10].

An important problem in hybrid systems theory is that of reachability analysis. In general terms, a reachability analysis problem consists in evaluating whether a given system will reach certain unsafe states, starting from certain initial states. This problem is associated with the safety verification problem: if the system cannot reach any unsafe state, then the system is declared to be safe. In the probabilistic setting, the safety verification problem can be formulated as that of checking whether the probability that the system trajectories reach an unsafe state from its initial states can be bounded by some given probability threshold.

In this paper, we focus on the probabilistic hybrid automata model [6], an extension of hybrid automata where the jumps involve probability distributions. This makes it possible to model component failures, message losses, buffer overflows and the like. Since these phenomena are important aspects when aiming at faithful models for networked and embedded applications, the interest in this formalism is growing [11,12].

Up to now, foundational results on the probabilistic reachability problem for probabilistic hybrid automata are scarce. Since they form a strict superclass of hybrid automata, this is not surprising. Decidability results are known for probabilistic linear hybrid automata and o-minimal hybrid automata [6].

This paper reports how we harvest and combine recent advances in the hybrid automata and the probabilistic automata worlds, in order to treat the general case. We are doing so by computing safe over-approximations via abstractions in the continuous as well as the probabilistic domain. One of the core challenges then is how to construct a sound probabilistic abstraction over a given covering of the state space. For this purpose, we first consider the non-probabilistic hybrid automaton obtained by replacing probabilistic branching with nondeterministic choices. Provided that there is a finite abstraction for this classical hybrid automaton, we then decorate this abstraction with probabilities to obtain a probabilistic abstraction, namely a finite probabilistic automaton [13]. We show the soundness of this abstraction, which allows us to verify probabilistic safety properties on the abstraction: if such a property holds in the abstraction, it holds also in the concrete system. Otherwise, refinement of the abstraction is required to obtain a more precise result.

Our abstraction approach can be considered as an orthogonal combination of the abstraction for hybrid automata [4,14], and Markov decision processes [15,16]. Because of this orthogonality, abstractions of probabilistic hybrid automata can be computed via abstractions for non-probabilistic hybrid automata and Markov decision processes. To show the applicability of this combination, we implemented a prototype tool, `ProHVer`, that first builds an abstraction via existing techniques [17] for classical hybrid automata, and then via techniques for Markov decision processes [15,16,18]. Subsequently, a fixed-point engine computes the reachability probabilities on the abstraction, which provides a safe upper bound. If needed, iterative refinement of the hybrid abstraction is performed. We report several successful applications of this prototypical implementation on different case studies. To the best of our knowledge, this is the first implementation which automatically checks safety properties for probabilistic hybrid automata.

2 Related Work

The verification of safety properties is undecidable for general hybrid automata. However, certain classes (e.g., initialised rectangular automata [19], o-minimal hybrid automata [20]), are decidable, and there are algorithms that construct finite bisimulation quotient automata. These results have been lifted to probabilistic hybrid automata [6], and provide exact results, rooted in a bisimulation-based abstraction. In these special cases, our approach can yield the same results, but it gives us the freedom to use different abstractions, that are more adapted to the problem at hand, but then not exact, but over-approximating. We actually treat the general case using that a practical verification can —to a certain extent— circumvent the decidability barrier by a semi-decision algorithm: we exploit tools that can, in practice, verify hybrid automata belonging to undecidable classes, to verify corresponding probabilistic hybrid automata.

The abstraction approach has also successfully been applied to probabilistic timed automata [18,21], a class of probabilistic hybrid automata, where only derivatives of constant value 1 occur. Their abstract analysis is based on difference-bound matrices (DBMs), and does not extend to the general setting considered here. Fränzle et al. [11, 12] use stochastic SAT to solve reachability problems on probabilistic hybrid automata. Their analysis is limited to depth-bounded reachability properties, i.e., the probability of reaching a location within at most N discrete jumps.

While the model we consider has probabilistic discrete jumps, there are several other suggestions equipping hybrid automata with continuous-time jumps. Davis [22] introduced piecewise deterministic Markov processes, whose state changes are triggered spontaneously as in continuous-time Markov chains. Stochastic differential equations [23] incorporate the continuous dynamics with random perturbations, such as Brownian motion. In stochastic hybrid systems [24,25], the transitions between different locations are resolved via a race between different Poisson processes. While these models enjoy a variety of applications, their analysis are limited and often based on Monte-Carlo simulations [8,9,26,27].

3 Preliminaries

In this section, we repeat the definition of conventional hybrid automata, in the style of [4], followed by the definition of probabilistic hybrid automata [6].

3.1 Hybrid Automata

We fix a variable m ranging over a finite set of discrete modes $\mathtt{M} = \{m_1, \ldots, m_n\}$ and variables $x_1, \ldots, x_k$ ranging over reals $\mathbb{R}$. We denote by S the resulting state space $\mathtt{M} \times \mathbb{R}^k$. For denoting the derivatives of $x_1, \ldots, x_k$ we use variables $\dot{x}_1, \ldots, \dot{x}_k$, ranging over $\mathbb{R}$ correspondingly. For simplicity, we sometimes use the vector $\boldsymbol{x}$ to denote $(x_1, \ldots, x_k)$, and $(m, \boldsymbol{x})$ to denote a state. Similar notations are used for the primed and dotted versions $\boldsymbol{x}'$, $\dot{\boldsymbol{x}}$.

In order to describe hybrid automata we use constraints that are arbitrary Boolean combinations of equalities and inequalities over terms. These constraints are used, on the one hand, to describe the possible flows and jumps and, on the other hand, to mark certain parts of the state space (e.g., the set of initial/unsafe states). A *state space constraint* is a constraint over the variables $m, \boldsymbol{x}$. A *flow constraint* is a constraint over the variables m, $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$.

For capturing the jump behaviours, we introduce the notion of update constraints. An *update constraint* $\mathtt{u}$, also called a *guarded command*, has the form: $\mathit{condition} \rightarrow \mathit{update}$ where $\mathit{condition}$ is a constraint over $m, \boldsymbol{x}$, and update is an expression denoting a function $\mathtt{M} \times \mathbb{R}^k \rightarrow \mathtt{M} \times \mathbb{R}^k$ which is called the reset mapping for m and $\boldsymbol{x}$. Intuitively, assume that the state $(m, \boldsymbol{x})$ satisfies $\mathit{condition}$, then the mode m and variable $\boldsymbol{x}$ are updated[1] to the new state $\mathit{update}(m, \boldsymbol{x})$.

A *jump constraint* is a finite disjunction $\bigvee_{\mathtt{u} \in \mathtt{U}} \mathtt{u}$ where $\mathtt{U}$ is a set of guarded commands. The constraint $\bigvee_{\mathtt{u} \in \mathtt{U}} \mathtt{u}$ can be represented by the set $\mathtt{U}$ for simplicity.

A *hybrid automaton* is a tuple $\mathcal{H} = (\mathit{Flow}, \mathtt{U}, \mathit{Init}, \mathit{UnSafe})$ consisting of a flow constraint Flow, a finite set of update constraints $\mathtt{U}$, a state space constraint Init describing the set of initial states, and a state space constraint UnSafe describing the set of unsafe states.

A *flow* of length l in a mode m is a function $r : [0, l] \mapsto \mathbb{R}^k$ with $l > 0$ such that r is differentiable for all $t \in [0, l]$, and for all $t \in [0, l]$, $(m, r(t), \dot{r}(t))$ satisfies Flow, where $\dot{r}$ is the derivative of r.

Transition System Semantics. The semantics of a hybrid automaton is a transition system with an uncountable set of states. Formally, the semantics of $\mathcal{H} = \{\mathit{Flow}, \mathtt{U}, \mathit{Init}, \mathit{UnSafe}\}$ is a transition system $T_{\mathcal{H}} = (S, T, S_{\mathit{Init}}, S_{\mathit{UnSafe}})$ where $S = \mathtt{M} \times \mathbb{R}^k$ is the set of states, $S_{\mathit{Init}} = \{s \in S \mid s \text{ satisfies } \mathit{Init}\}$ denotes the set of initial states, and $S_{\mathit{UnSafe}} = \{s \in S \mid s \text{ satisfies } \mathit{UnSafe}\}$ represents the set of unsafe states. The transition set T is defined as the union of two transition relations $T_{\mathcal{C}}, T_{\mathcal{D}} \subseteq S \times S$, where $T_{\mathcal{C}}$ corresponds to transitions due to continuous flows defined by:

– $((m, \boldsymbol{x}), (m, \boldsymbol{x'})) \in T_{\mathcal{C}}$, if there exists a flow r of length l in m such that $r(0) = \boldsymbol{x}$ and $r(l) = \boldsymbol{x'}$;

and $T_{\mathcal{D}}$ corresponds to transitions due to discrete jumps. The transition due to an update constraint $\mathtt{u} : \mathit{condition} \rightarrow \mathit{update}$, denoted by $T_{\mathcal{D}}(\mathtt{u})$ is defined by:

– $((m, \boldsymbol{x}), (m', \boldsymbol{x'})) \in T_{\mathcal{D}}(\mathtt{u})$ if $(m, \boldsymbol{x})$ satisfies the guard $\mathit{condition}$ and it holds that $(m', \boldsymbol{x'}) = \mathit{update}(m, \boldsymbol{x})$.

Then, we define $T_{\mathcal{D}} = \cup_{\mathtt{u} \in \mathtt{U}} T_{\mathcal{D}}(\mathtt{u})$.

In the rest of the paper, if no confusion arises, we use *Init* to denote both the constraint for the initial states and the set of initial states. Similarly, UnSafe is used to denote both the constraint for the unsafe states and the set of unsafe states.

3.2 Probabilistic Automata

For defining the semantics of a probabilistic hybrid automaton, we recall first the notion of a probabilistic automaton [13]. It is an extension of a transition system with probabilistic branching.

[1] Our definition of jumps is deterministic, as in [14], i.e., if a jump is triggered for a state satisfying $\mathit{condition}$, the successor state is updated deterministically according to update. In [4], the jump is defined to be nondeterministic: if a state satisfies $\mathit{condition}$, a successor will be selected nondeterministically from a set of states. Our method can be easily extended to this. We restrict to deterministic jumps for simplicity of the presentation in this paper.

We first introduce some notation. Let S be a (possibly uncountable) set. A *distribution* over S is a function $\mu : S \to [0, 1]$ such that (a) the set $\{s \in S \mid \mu(s) > 0\}$ is finite, and (b) the sum $\sum_{s \in S} \mu(s) = 1$. Let the support $Supp(\mu)$ of μ be $\{s \in S \mid \mu(s) > 0\}$. Let $Distr(S)$ denote the set of all distributions over S. For an arbitrary but fixed state s in S, a *Dirac distribution* for s, denoted by $Dirac_s$, is a distribution over S such that $Dirac_s(s) = 1$, that is, $Supp(Dirac_s) = \{s\}$. Note that the Dirac distribution will be used to describe the continuous evolution of a probabilistic hybrid automaton.

Definition 1. *A probabilistic automaton $\mathcal{M}$ is a tuple $(S, Steps, Init, UnSafe)$, where $Steps \subseteq S \times Distr(S)$, $Init \subseteq S$, and $UnSafe \subseteq S$. Here, S denotes the (possible uncountable) set of states, $Init$ is the set of initial states, $UnSafe$ the set of unsafe states, and $Steps \subseteq S \times Distr(S)$ the transition relation.*

For a transition $(s, \mu) \in Steps$, we use $s \to \mu$ as a shorthand notation, and call μ a successor distribution of s. Let $Steps(s)$ be the set $\{\mu \mid (s, \mu) \in Steps\}$. We assume that $Steps(s) \neq \emptyset$ for all $s \in S$.

A path of $\mathcal{M}$ is a finite or infinite sequence $\sigma = s_0\mu_0s_1\mu_1 \ldots$ such that $s_i \to \mu_i$ and $\mu_i(s_{i+1}) > 0$ for all possible $i \geq 0$. We denote by $first(\sigma)$ the first state s_0 of σ, by $\sigma[i]$ the $i + 1$-th state s_i, and, if σ is finite, by $last(\sigma)$ the last state of σ. Let $Path$ be the set of all infinite paths and $Path^*$ the set of all finite paths.

The non-deterministic choices in $\mathcal{M}$ can be resolved by *adversaries*. Formally, an adversary of $\mathcal{M}$ is a map $A : Path^* \to Distr(Steps)$ such that $A(\sigma)(s, \mu) > 0$ implies that $s = last(\sigma)$ and $s \to \mu$. Intuitively, if $A(\sigma)(s, \mu) > 0$, then the successor distribution μ should be selected from state s with probability $A(\sigma)(s, \mu)$. Moreover, an adversary A is called *Markovian* if for all $\sigma \in Path^*$, $A(\sigma) = A(last(\sigma))$, that is, for each finite path, A depends only on its last state. An adversary A is called *deterministic* if for all $\sigma \in Path^*$, $A(\sigma)$ is always a Dirac distribution. We say that an adversary A is *simple* if A is Markovian and deterministic. Given an adversary A and an initial state s, a unique probability measure over $Path$, which is denoted by $Prob_s^A$, can be defined.

3.3 Probabilistic Hybrid Automata

Now we recall the definition of probabilistic hybrid automata, by equipping the discrete jumps with probabilities. This is needed to model, for example, component failure or message losses.

For capturing the probabilistic jump behaviours, a *guarded command* c is defined to have the form

$$condition \to p_1 : update_1 + \ldots + p_{q_\mathsf{c}} : update_{q_\mathsf{c}}$$

where $q_\mathsf{c} \geq 1$ denotes the number of probabilistic branching of c, $p_i > 0$ for $i = 1, \ldots, q_\mathsf{c}$ and $\sum_{i=1}^{q_\mathsf{c}} p_i = 1$, *condition* is a constraint over $(m, \boldsymbol{x})$, and $update_i$ is an expression denoting a reset mapping for m and $\boldsymbol{x}$ for all $i = 1, \ldots, q_\mathsf{c}$. Intuitively, if a state $(m, \boldsymbol{x})$ satisfies the guard *condition*, a jump to states $(m_1, \boldsymbol{x}_1), \ldots, (m_{q_\mathsf{c}}, \boldsymbol{x}_{q_\mathsf{c}})$ occurs such that $(m_i, \boldsymbol{x}_i) = update_i(m, \boldsymbol{x})$ is selected with probability p_i for $i = 1, \ldots, q_\mathsf{c}$. Observe that for different $i \neq j$, it could be the case that $(m_i, \boldsymbol{x_i}) = (m_j, \boldsymbol{x_j})$. In this paper we assume that q_c is finite for all c.

Definition 2. *A* probabilistic hybrid automaton *is a tuple $\mathcal{H} = (\mathit{Flow}, \mathsf{C}, \mathit{Init}, \mathit{UnSafe})$ where Flow, Init, UnSafe are the same as in the hybrid automaton, and C is a finite set of guarded commands c.*

The probabilistic hybrid automaton induces a classical hybrid automaton where probabilistic branching is replaced by nondeterministic choices. Intuitively, the semantics of the latter spans the semantics of the former.

Definition 3. *Let $\mathsf{c} : \mathit{condition} \to p_1 : \mathit{update}_1 + \ldots + p_{q_c} : \mathit{update}_{q_c}$ be a guarded command. It induces a set of q update constraints: $\mathit{ind}(\mathsf{c}) = \{\mathsf{u}_1, \ldots, \mathsf{u}_{q_c}\}$ where u_i corresponds to the update constraint $\mathit{condition} \to \mathit{update}_i$ for $i = 1, \ldots, q_c$. Moreover, we define $\mathit{ind}(\mathsf{C}) := \bigcup_{\mathsf{c}\in\mathsf{C}} \mathit{ind}(\mathsf{c})$.*

Let $\mathcal{H} = (\mathit{Flow}, \mathsf{C}, \mathit{Init}, \mathit{UnSafe})$ be a probabilistic hybrid automaton. The induced hybrid automaton is a tuple $\mathit{ind}(\mathcal{H}) = (\mathit{Flow}, \mathit{ind}(\mathsf{C}), \mathit{Init}, \mathit{UnSafe})$.

Semantics. The semantics of a probabilistic hybrid automaton is a probabilistic automaton [6]. Let $\mathcal{H} = (\mathit{Flow}, \mathsf{C}, \mathit{Init}, \mathit{UnSafe})$ be a probabilistic hybrid automaton. Let $\mathit{ind}(\mathcal{H})$ denote the induced hybrid automaton, and let $T_{\mathit{ind}(\mathcal{H})} = (S, T, \mathit{Init}, \mathit{UnSafe})$ denote the transition system representing the semantics of $\mathit{ind}(\mathcal{H})$. Recall that $T = T_{\mathcal{C}} \cup T_{\mathcal{D}}$ where $T_{\mathcal{C}}$ corresponds to transitions due to continuous flow and $T_{\mathcal{D}}$ corresponds to transitions due to discrete jumps.

The semantics of $\mathcal{H}$ is the probabilistic automaton $\mathcal{M}_{\mathcal{H}} = (S, \mathit{Steps}, \mathit{Init}, \mathit{UnSafe})$ where $S, \mathit{Init}, \mathit{UnSafe}$ are the same as in $T_{\mathit{ind}(\mathcal{H})}$, and Steps is defined as the union of two transition relations $\mathit{Steps}_{\mathcal{C}}, \mathit{Steps}_{\mathcal{D}} \subseteq S \times \mathit{Distr}(S)$. Here, as in the non-probabilistic setting, $\mathit{Steps}_{\mathcal{C}}$ corresponds to transitions due to continuous flows, while $\mathit{Steps}_{\mathcal{D}}$ corresponds to transitions due to discrete jumps. Both of them are defined respectively as follows.

For each transition $((m, \boldsymbol{x}), (m, \boldsymbol{x}')) \in T_{\mathcal{C}}$ in $\mathit{ind}(\mathcal{H})$, there is a corresponding transition in $\mathcal{H}$ from $(m, \boldsymbol{x})$ to $(m, \boldsymbol{x}')$ with probability 1. So, $\mathit{Steps}_{\mathcal{C}}$ is defined by: $\mathit{Steps}_{\mathcal{C}} = \{((m, \boldsymbol{x}), \mathit{Dirac}_{(m,\boldsymbol{x}')}) \mid ((m, \boldsymbol{x}), (m, \boldsymbol{x}')) \in T_{\mathcal{C}}\}$.

Now we discuss transitions induced by discrete jumps. First, for a guarded command c, we define the set $\mathit{Steps}_{\mathcal{D}}(\mathsf{c})$ corresponding to it. Let $\mathit{ind}(\mathsf{c}) = \{\mathsf{u}_1, \ldots, \mathsf{u}_{q_c}\}$ be as defined in Definition 3. Then, for arbitrary $q_{\mathsf{c}} + 1$ states $(m, \boldsymbol{x})$, $(m_1, \boldsymbol{x}_1), \ldots, (m_{q_c}, \boldsymbol{x}_{q_c}) \in S$ satisfying the condition $((m, \boldsymbol{x}), (m_i, \boldsymbol{x}_i)) \in T_{\mathcal{D}}(\mathsf{u}_i)$ for $i = 1, \ldots, q_{\mathsf{c}}$, we introduce the transition $((m, \boldsymbol{x}), \mu) \in \mathit{Steps}_{\mathcal{D}}(\mathsf{c})$ with

$$\mu(m_i, \boldsymbol{x}_i) = \sum_{j\in\{j \mid m_j = m_i \wedge \boldsymbol{x}_j = \boldsymbol{x}_i\}} p_j, \tag{1}$$

for $i = 1, \ldots, q_{\mathsf{c}}$. Then, $\mathit{Steps}_{\mathcal{D}}$ is defined to be $\bigcup_{\mathsf{c}\in\mathsf{C}} \mathit{Steps}_{\mathcal{D}}(\mathsf{c})$. Recall that we have assumed that q_{c} is finite for all c. This implies $\mathit{Supp}(\mu)$ is finite for all transitions (s, μ) with $s \in S$.

Safety Properties. For hybrid automata, the safety property asserts that the unsafe states can never be reached. For probabilistic hybrid automata, however, the safety property expresses that the maximal probability of reaching the set UnSafe is bounded by some give threshold ε. In the following we fix a certain threshold ε. Let $\mathit{Reach}(\mathit{UnSafe})$ denote the set of paths $\{\sigma \in \mathit{Path} \mid \exists i.\ \sigma[i] \in \mathit{UnSafe}\}$. The automaton $\mathcal{H}$ is called *safe* if for each adversary A and each initial state s of $\mathcal{M}(\mathcal{H})$, $Prob_s^A(Reach(UnSafe)) \leq \varepsilon$ holds. In this paper, we would like to develop a framework to deal with such a probabilistic safety verification problem for general probabilistic hybrid automata.

Simulation Relations. We recall the notion of simulations between probabilistic automata. Intuitively, if $\mathcal{M}_2$ simulates $\mathcal{M}_1$, that is, $\mathcal{M}_2$ is an over-approximation of $\mathcal{M}_1$, then $\mathcal{M}_2$ can mimic all behaviours of $\mathcal{M}_1$. Thus, this allows us to verify safety properties on the abstraction $\mathcal{M}_2$ instead of $\mathcal{M}_1$. To establish the notion of simulations, we introduce first the notion of weight functions [28], which establish the correspondence between distributions.

Definition 4. *Let $\mu_1 \in Distr(S_1)$ and $\mu_2 \in Distr(S_2)$ be two distributions. For a relation $R \subseteq S_1 \times S_2$, a weight function for (μ_1, μ_2) with respect to R is a function $\Delta : S_1 \times S_2 \to [0,1]$ such that (i) $\Delta(s_1, s_2) > 0$ implies $(s_1, s_2) \in R$, (ii) $\mu_1(s_1) = \sum_{s_2 \in S_2} \Delta(s_1, s_2)$ for $s_1 \in S_1$, and (iii) $\mu_2(s_2) = \sum_{s_1 \in S_1} \Delta(s_1, s_2)$ for $s_2 \in S_2$.*

We write $\mu_1 \sqsubseteq_R \mu_2$ if and only if there exists a weight function for μ_1 and μ_2 with respect to R.

Now, we recall the notion of simulations [13]. The simulation requires that every successor distribution of a state of $\mathcal{M}_1$ is related to a successor distribution of its corresponding state of $\mathcal{M}_2$ via a weight function.

Definition 5. *Given two automata $\mathcal{M}_1 = (S_1, Init_1, Steps_1, UnSafe_1)$ and $\mathcal{M}_2 = (S_2, Init_2, Steps_2, UnSafe_2)$, we say that $\mathcal{M}_2$ simulates $\mathcal{M}_1$, denoted by $\mathcal{M}_1 \preceq \mathcal{M}_2$, if and only if there exists a relation $R \subseteq S_1 \times S_2$, which we will call simulation relation from now on, such that*

1. *for each $s_1 \in Init_1$ there exists an $s_2 \in Init_2$ with $(s_1, s_2) \in R$.*
2. *for each $s_1 \in UnSafe_1$ there exists an $s_2 \in UnSafe_2$ with $(s_1, s_2) \in R$.*
3. *for each pair $(s_1, s_2) \in R$, if there exists $(s_1, \mu_1) \in Steps_1$, there exists a distribution $\mu_2 \in Distr(S_2)$ such that $(s_2, \mu_2) \in Steps_2$ and $\mu_1 \sqsubseteq_R \mu_2$.*

4 Abstractions for Probabilistic Hybrid Automata

Various abstraction refinement techniques have been developed for verifying safety properties against non-probabilistic hybrid automata. All of them have a common strategy: the set S is covered by a finite set of abstract states, each representing a set of concrete states. Then, the abstraction is constructed which is an over-approximation of the original system. Afterwards, the safety property is checked on the abstraction. If the set of unsafe states is unreachable, the original system is safe since the abstraction over-approximates the original system. If not, the covering might have been chosen too coarse, and a refinement step is needed. Based on this idea, predicate abstraction based abstraction refinement has been used [3,14] for safety verification of linear hybrid automata, and constraint propagation based abstraction refinement has been used for safety verification of general hybrid automata [4].

Let $\mathcal{H} = (Flow, \mathtt{C}, Init, UnSafe)$ be a probabilistic hybrid automaton. The aim of this section is—independent of which abstraction technique is used—to develop a framework for constructing an abstraction for $\mathcal{H}$, which is a finite probabilistic automaton. First we introduce the notion of abstract states which form a (not necessarily disjoint) covering of the concrete state space:

Definition 6. *An abstract state is a pair (m, B) where $m \in \mathcal{M}$ and $B \subseteq \mathbb{R}^k$. The set $\mathcal{B}$ is a finite set of abstract states such that $S = \bigcup\{(m, \boldsymbol{x}) \mid (m, B) \in \mathcal{B} \wedge \boldsymbol{x} \in B\}$.*

In the above definition, any two abstract states (m, B_1) and (m, B_2) may have common interiors, including common borders[2]. The case allowing common interiors is the case if the polyhedra based abstraction technique is used [17], and common border is the case if the constraint propagation based abstraction technique is used [4]. Our abstraction scheme in this section works for all of them.

Fig. 1 illustrates how this section is organised. Given a probabilistic hybrid automaton $\mathcal{H}$ and an abstract state space $\mathcal{B}$, we introduce the quotient automaton for both $ind(\mathcal{H})$ and $\mathcal{H}$ in Sec. 4.1, respectively. In Sec. 4.2, we show the soundness with respect to the quotient automaton (cf. Lemma 1 and Lemma 2).

The quotient automaton is in general hard to compute. Thus, we introduce in Sec. 4.3 general abstractions, which over-approximate the quotient automata conservatively. In Sec. 4.4, we discuss how the abstraction for the given probabilistic hybrid automaton is constructed (see Fig. 1): we construct first the abstraction of the induced hybrid automaton, from which the abstraction of the probabilistic setting is then obtained.

$$\begin{tikzcd}
\mathcal{H} \arrow[r] & ind(\mathcal{H}) \arrow[d] \\
Quo_{\mathcal{H}}(\mathcal{B}) \arrow[u, dotted, "\text{Lemma 1}\ \text{Lemma 2}"'] & Quo_{ind(\mathcal{H})}(\mathcal{B}) \arrow[d] \\
Abs_{\mathcal{H}}(\mathcal{B}) \arrow[u, dotted, "\text{Lemma 3}\ \text{Theorem 1}"'] & Abs_{ind(\mathcal{H})}(\mathcal{B}) \arrow[l]
\end{tikzcd}$$

Fig. 1. Computation of the abstraction

4.1 Quotient Automaton for $\mathcal{H}$

We define the quotient automaton for the probabilistic hybrid automaton $\mathcal{H}$. First we define the quotient automaton for the induced hybrid automaton $ind(\mathcal{H})$. As a convention we use $\mathcal{T}, \mathcal{I}, \mathcal{U}$ to denote the set of transitions, initial states, unsafe states in the quotient automata.

Definition 7. *Let $\mathcal{H} = (Flow, \mathcal{C}, Init, UnSafe)$ be a probabilistic hybrid automaton, and let $\mathcal{B}$ denote the abstract state space. Let $T_{ind(\mathcal{H})} = (S, T_{\mathcal{C}} \cup T_{\mathcal{D}}, Init, UnSafe)$ denote the automaton representing the semantics of $ind(\mathcal{H})$. The quotient automaton for $T_{ind(\mathcal{H})}$, denoted by $Quo_{ind(\mathcal{H})}(\mathcal{B})$, is a finite transition system $(\mathcal{B}, \mathcal{T}, \mathcal{I}, \mathcal{U})$ where*

- $\mathcal{I} = \{(m, B) \in \mathcal{B} \mid \exists \boldsymbol{x} \in B.\ (m, \boldsymbol{x}) \in Init\}$,
- $\mathcal{U} = \{(m, B) \in \mathcal{B} \mid \exists \boldsymbol{x} \in B.\ (m, \boldsymbol{x}) \in UnSafe\}$,
- *$\mathcal{T}_{\mathcal{C}}$ corresponds to the set of abstract transitions due to continuous flow:* $\mathcal{T}_{\mathcal{C}} = \{((m, B), (m, B')) \in \mathcal{B}^2 \mid \exists \boldsymbol{x} \in B \wedge \exists \boldsymbol{x'} \in B' \wedge ((m, \boldsymbol{x}), (m, \boldsymbol{x'})) \in T_{\mathcal{C}}\}$,
- *$\mathcal{T}_{\mathcal{D}}$ corresponds to the set of abstract transitions due to discrete jumps. We first define the transition induced by one fixed update $u \in ind(\mathcal{C})$. Assume that we have*

[2] We may also require that abstract states form a partitioning over the original state S, with pairwise disjoint abstract states. Such abstractions are, however, harder to construct for non-trivial models.

$((m, \boldsymbol{x}), (m', \boldsymbol{x'})) \in Steps_{\mathcal{D}}(u)$. Then, it induces an abstract transition $((m, B),$ $(m', B')) \in \mathcal{T}_{\mathcal{D}}(u)$ where B, B' are the abstract states containing $\boldsymbol{x}$, $\boldsymbol{x'}$ respectively. Then, let $\mathcal{T}_{\mathcal{D}} = \cup_{u \in ind(\mathcal{C})} \mathcal{T}_{\mathcal{D}}(u)$.

Let $\mathcal{H} = (Flow, \mathsf{C}, Init, UnSafe)$ be a probabilistic hybrid automaton, and let $\mathcal{M}_{\mathcal{H}} = (S, Steps_{\mathcal{C}} \cup Steps_{\mathcal{D}}, Init, UnSafe)$ denote the automaton representing the semantics of $\mathcal{H}$. As in the induced non-probabilistic setting, we define a quotient automaton, denoted by $Quo_{\mathcal{H}}(\mathcal{B})$, for an abstract state space $\mathcal{B}$. For this we first introduce the set of lifted distributions:

Definition 8. *Let $\mathcal{H}$ and $\mathcal{M}_{\mathcal{H}}$ be as described above. Let $\mathcal{B}$ denote the abstract state space. Let $c \in \mathsf{C}$ and assume that $(s, \mu) \in Steps_{\mathcal{D}}(c)$ in $\mathcal{M}_{\mathcal{H}}$. By definition of $Steps_{\mathcal{D}}(c)$, there exist states $(m_1, \boldsymbol{x}_1), \ldots, (m_{q_c}, \boldsymbol{x}_{q_c}) \in S$ satisfying the condition $((m, \boldsymbol{x}), (m_i, \boldsymbol{x}_i)) \in T_{\mathcal{D}}(u_i)$ for $i = 1, \ldots, q_c$. Then, for arbitrary abstract states $(m_1, B_1), \ldots, (m_{q_c}, B_{q_c})$ with $\boldsymbol{x_i} \in B_i$ for $i = 1, \ldots, q_c$ we introduce the distribution $\mu' \in Distr(\mathcal{B})$ by: $\mu'(m_i, B_i) = \sum_{\{j \mid (m_j, B_j) = (m_i, B_i)\}} \mu(m_j, \boldsymbol{x_j})$. The set of lifted distributions $lift_{\mathcal{B}}(\mu)$ contains all such μ'.*

Let μ be the distribution according to a guarded command c. Since the covering $\mathcal{B}$ is in general not disjoint, a concrete state $(m_i, \boldsymbol{x_i})$ might belong to more than one abstract states. In this case μ induces more than one lifted distribution. In the above definition, this is reflected by the way of defining one specific lifted distribution μ', for which we first fix to which abstract state each concrete state $(m_i, \boldsymbol{x_i})$ belongs. Note that if $\mathcal{B}$ is a disjoint partitioning of S, the set $lift_{\mathcal{B}}(\mu)$ is a singleton. We now introduce the quotient automaton for the probabilistic hybrid automaton:

Definition 9. *Let $\mathcal{H}$ and $\mathcal{M}_{\mathcal{H}}$ be as described above. Let $\mathcal{B}$ denote the abstract state space. The quotient automaton for $\mathcal{M}_{\mathcal{H}}$ with respect to $\mathcal{B}$ is defined by $Quo_{\mathcal{H}}(\mathcal{B}) = (\mathcal{B}, \mathcal{ST}, \mathcal{I}, \mathcal{U})$ where $\mathcal{I}$ and $\mathcal{U}$ are defined as for $Quo_{ind(\mathcal{H})}(\mathcal{B})$, and $\mathcal{ST} = \mathcal{ST}_{\mathcal{C}} \cup \mathcal{ST}_{\mathcal{D}}$ is the set of abstract transitions where:*

- *$\mathcal{ST}_{\mathcal{C}}$ corresponds to the set of abstract transitions due to continuous flow: $\mathcal{ST}_{\mathcal{C}} = \{((m, B), Dirac_{(m,B')}) \mid \exists \boldsymbol{x} \in B \wedge \exists \boldsymbol{x'} \in B' \wedge ((m, \boldsymbol{x}), (m, \boldsymbol{x'})) \in Steps_{\mathcal{C}}\}$.*
- *$\mathcal{ST}_{\mathcal{D}}$ corresponds to the set of abstract transitions due to discrete jumps. We first define the transition induced by one fixed guarded command c. Consider all $((m, \boldsymbol{x}), \mu) \in Steps_{\mathcal{D}}(c)$. These pairs induce corresponding abstract transitions $((m, B), \mu') \in \mathcal{ST}_{\mathcal{D}}(c)$ where B is the abstract state containing $\boldsymbol{x}$, and $\mu' \in lift_{\mathcal{B}}(\mu)$. Then, let $\mathcal{ST}_{\mathcal{D}} = \cup_{c \in \mathsf{C}} \mathcal{ST}_{\mathcal{D}}(c)$.*

Example 1. Consider Fig. 2 and assume we have a guarded command $\mathsf{c} : condition \rightarrow p_1 : up_1 + \ldots + p_4 : up_4$. Thus $q_{\mathsf{c}} = 4$. The abstract states are represented by circles, labelled with the corresponding tuple. The concrete states are represented by black points, labelled with only the evaluation of the variables (assume that all of them are different). Thus s_0 represents state (m_0, s_0) and so on. Arrows are transitions in the concrete models, where the labels represent the probability p_i of the corresponding update up_i of c.

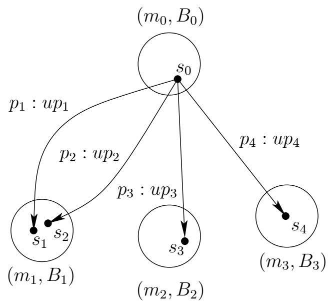

Fig. 2. Illustrating the abstract discrete transitions in the quotient automaton

Consider the two (there may be more) concrete transitions in $T_{ind(\mathcal{H})}$: $((m_0, s_0), (m_1, s_1))$, $((m_0, s_0), (m_1, s_2)) \in T_{\mathcal{D}}$. Both of them lead from (m_0, B_0) to the same abstract state (m_1, B_1). By Definition 7, we have that $((m_0, B_0), (m_i, B_i)) \in \mathcal{T}_{\mathcal{D}}$ for $i = 1, 2, 3$ in $Quo_{ind(\mathcal{H})}(\mathcal{B})$.

We have a concrete transition $((m_0, s_0), \mu)$ where μ is defined by: $\mu(s_i) = p_i$ for $i = 1, 2, 3, 4$. Assume first that B_0, B_1, B_2, B_3 are disjoint. By Definition 8, $lift_{\mathcal{B}}(\mu) = \{\mu'\}$ where μ' is defined by: $\mu'(m_1, B_1) = p_1 + p_2$, $\mu'(m_2, B_2) = p_3$, and $\mu'(m_3, B_3) = p_4$. Then, by Definition 9, this induces an abstract transition $((m_0, B_0), \mu') \in lift_{\mathcal{B}}(\mu)$ in $Quo_{\mathcal{H}}(\mathcal{B})$.

Assume now that the abstract states B_1 and B_2 are not disjoint, and that s_2 is on the common border of (m_1, B_1) and (m_2, B_2) (which implies also $m_1 = m_2$). In this case the set $lift_{\mathcal{B}}(\mu)$ contains another element μ'' which defined by: $\mu''(m_1, B_1) = p_1$, $\mu''(m_2, B_2) = p_2 + p_3$ and $\mu''(m_3, B_3) = p_4$. Again by Definition 9, μ'' induces another abstract transition $((m_0, B_0), \mu'')$ in $Quo_{\mathcal{H}}(\mathcal{B})$.

4.2 Soundness

Given a probabilistic hybrid automaton $\mathcal{H}$ and a set of abstract states $\mathcal{B}$, we defined a probabilistic quotient automaton $Quo_{\mathcal{H}}(\mathcal{B})$. The following lemma shows that this automaton conservatively over-approximates $\mathcal{M}_{\mathcal{H}}$.

Lemma 1. *$Quo_{\mathcal{H}}(\mathcal{B})$ simulates $\mathcal{M}_{\mathcal{H}}$.*

Proof sketch: We define $R = \{((m, \boldsymbol{x}), (m', B)) \in S \times \mathcal{B} \mid m = m' \wedge \boldsymbol{x} \in B\}$. It suffices to show that R is a simulation relation. Let $((m, \boldsymbol{x}), (m, B)) \in R$. The first two conditions for simulation relations are trivially satisfied. It remains the third condition. There are two type of transitions starting from $(m, \boldsymbol{x})$ in $\mathcal{M}_{\mathcal{H}}$: the case $((m, \boldsymbol{x}), Dirac_{(m,\boldsymbol{x}')}) \in Steps_{\mathcal{C}}$ is trivial and skipped. Now consider the case $((m, \boldsymbol{x}), \mu) \in Steps_{\mathcal{D}}$: there exists then a guarded command c such that $((m, \boldsymbol{x}), \mu) \in Steps_{\mathcal{D}}(\mathrm{c})$. Let c and $ind(\mathrm{c}) = \{\mathrm{u}_1, \ldots, \mathrm{u}_{q_\mathrm{c}}\}$ be as described in Definition 3, and let $(m_i, \boldsymbol{x}_i) = update_i(m, \boldsymbol{x})$ be the state with respect to $update_i$ for $i = 1, \ldots, q_\mathrm{c}$. Note that it could be the case that, for $i \neq j$, $\boldsymbol{x_i} = \boldsymbol{x_j}$. Moreover, let $(m_i, B_i) \in \mathcal{B}$ denote the abstract state satisfying $\boldsymbol{x}_i \in B_i$. By construction of the relation R, we know that $((m_i, \boldsymbol{x_i}), (m_i, B_i)) \in R$. By the definition of $\mathcal{ST}$ (cf. Definition 11), we have that $((m, B), \mu') \in \mathcal{ST}_{\mathcal{D}}(\mathrm{c})$ where $\mu'(m_i, B_i) = \sum_{j \in \{j \mid m_j = m_i \wedge B_j = B_i\}} p_j$ for $i = 1, \ldots, q_\mathrm{c}$. Define Δ for (μ, μ') with respect to R by: $\Delta((m_i, \boldsymbol{x}_i), (m_i, B_i))$ equals $\mu(m_i, \boldsymbol{x}_i)$ for $i = 1, \ldots, q_\mathrm{c}$, and equals 0 otherwise. It remains to show that Δ is the proper weight function. For the first condition, assume $\Delta((m^*, \boldsymbol{x}^*), (m', B')) > 0$. By the definition of Δ, we have $m^* = m'$ and $\boldsymbol{x}^* \in B'$, implying $((m^*, \boldsymbol{x}^*), (m', B')) \in R$. Now we show the third condition (the second condition is similar). Let (m_j, B_j) be an abstract state with $j \in \{1, \ldots, q_\mathrm{c}\}$ (otherwise trivial). On one hand, due to the definition of μ', $\mu'(m_j, B_j) = \sum_{i \in I} p_i$ where $I = \{i \mid m_i = m_j \wedge B_i = B_j\}$ denotes the set of all indices i such that $(m_i, B_i) = (m_j, B_j)$. On the other hand, by the definition of Δ, it holds $\sum_{i\in I} p_i = \sum_{\boldsymbol{x}_k \in B_j} \mu(m_j, \boldsymbol{x}_k) = \sum_{k\in I} \Delta((m_j, \boldsymbol{x}_k), (m_j, B_j))$ (cf. Equation (1)), which implies the third condition. ■

Since simulation on probabilistic automata preserves safety properties [13], we have the correctness of our construction:

Lemma 2. *The abstraction preserves the safety property: if the probability of reaching UnSafe in $Quo_{\mathcal{H}}(\mathcal{B})$ is bounded by ε, this is also the case in $\mathcal{H}$.*

4.3 Abstractions for $\mathcal{H}$

Consider the probabilistic hybrid automaton $\mathcal{H}$. Often the computation of the exact quotient automaton $Quo_{\mathcal{H}}(\mathcal{B})$ as defined in Definition 9 refers to concrete states, and is hard or even impossible. In this subsection we introduce the notion of abstractions which over-approximate the quotient automata. As a convention we use the primed version $\mathcal{T}', \mathcal{I}', \mathcal{U}'$ to denote the set of transitions, initial states, unsafe states in the abstraction.

Definition 10. *Let $\mathcal{H} = (Flow, \mathcal{C}, Init, UnSafe)$ be a probabilistic hybrid automaton, and let $\mathcal{B}$ denote the abstract state space. Then,*

- *$Abs_{ind(\mathcal{H})}(\mathcal{B}) = (\mathcal{B}, \mathcal{T}', \mathcal{I}', \mathcal{U}')$ is an abstraction of the quotient $Quo_{ind(\mathcal{H})}(\mathcal{B})$ iff $\mathcal{T}' = \cup_{u\in ind(\mathcal{C})} \mathcal{T}'_{\mathcal{D}}(u) \cup \mathcal{T}'_{\mathcal{C}}$ and it holds $\mathcal{T}_{\mathcal{C}} \subseteq \mathcal{T}'_{\mathcal{C}}$, $\mathcal{T}_{\mathcal{D}}(u) \subseteq \mathcal{T}'_{\mathcal{D}}(u)$ for $u \in ind(\mathcal{C})$, $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{U} \subseteq \mathcal{U}'$,*
- *$Abs_{\mathcal{H}}(\mathcal{B}) = (\mathcal{B}, \mathcal{ST}', \mathcal{I}', \mathcal{U}')$ is an abstraction of the quotient $Quo_{\mathcal{H}}(\mathcal{B})$ iff $\mathcal{ST}' = \cup_{c\in\mathcal{C}} \mathcal{ST}'_{\mathcal{D}}(c) \cup \mathcal{ST}'_{\mathcal{C}}$ and it holds $\mathcal{ST}_{\mathcal{D}}(c) \subseteq \mathcal{ST}'_{\mathcal{D}}(c)$ for $c \in \mathcal{C}$, $\mathcal{ST}_{\mathcal{C}} \subseteq \mathcal{ST}'_{\mathcal{C}}$, $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{U} \subseteq \mathcal{U}'$.*

In that case, we say also that $Abs_{ind(\mathcal{H})}(\mathcal{B})$ is an abstraction of the induced hybrid automaton $ind(\mathcal{H})$. Similarly, we say also that $Abs_{\mathcal{H}}(\mathcal{B})$ is an abstraction of the probabilistic hybrid automaton $\mathcal{H}$. Since the abstraction as defined may have more initial states, unsafe states and transitions than the quotient automaton, it is easy to verify that the abstraction simulates the corresponding quotient automaton. Since simulation is transitive, the abstraction also simulates the corresponding semantics automaton. Thus, the abstraction preserves also safety properties of $\mathcal{H}$.

4.4 Computing Abstractions

Let $\mathcal{H}$ be a probabilistic hybrid automaton. Existing methods can be used to compute an abstraction $Abs_{ind(\mathcal{H})}(\mathcal{B})$ for the induced hybrid automaton $ind(\mathcal{H})$, for example [4, 14,17]. In the following we define an abstraction based on $Abs_{ind(\mathcal{H})}(\mathcal{B})$:

Definition 11. *For a probabilistic hybrid automaton $\mathcal{H}$, let $\mathcal{B}$ be the abstract state space, and $Abs_{ind(\mathcal{H})} = (\mathcal{B}, \mathcal{T}'_{\mathcal{D}} \cup \mathcal{T}'_{\mathcal{C}}, \mathcal{I}', \mathcal{U}')$ be an abstraction of $ind(\mathcal{H})$. We define $Abs_{\mathcal{H}}(\mathcal{B}) = (\mathcal{B}, \mathcal{ST}'_{\mathcal{C}} \cup \mathcal{ST}'_{\mathcal{D}}, \mathcal{I}', \mathcal{U}')$ for $\mathcal{H}$ as follows:*

- *$\mathcal{ST}'_{\mathcal{C}} = \mathcal{T}'_{\mathcal{C}}$,*
- *$\mathcal{ST}'_{\mathcal{D}}$ corresponds to the set of abstract transitions due to discrete jumps. We first define the transition induced by one fixed guarded command c : $condition \rightarrow p_1 : update_1 + \ldots + p_{q_c} : update_{q_c}$, and $ind(c) = \{u_1, \ldots, u_{q_c}\}$ as defined in Definition 3. Then, for every sequence of abstract states*

$(m, B), (m_1, B_1), \ldots, (m_{q_c}, B_{q_c})$ satisfying the condition: $((m, B), (m_i, B_i)) \in \mathcal{T}'_{\mathcal{D}}(\mathrm{u}_i)$ for $i = 1, \ldots, q_c$ we introduce the transition $((m, B), \mu) \in \mathcal{ST}'_{\mathcal{D}}(c)$ such that $\mu(m_i, B_i) = \sum_{j \in \{j | m_j = m_i \wedge B_j = B_i\}} p_j$ for $i = 1, \ldots, q_c$. Then, $\mathcal{ST}'_{\mathcal{D}}$ is defined to be $\bigcup_{c \in C} \mathcal{ST}'_{\mathcal{D}}(c)$.

Is $Abs_{\mathcal{H}}(\mathcal{B})$ in fact an abstraction of $\mathcal{H}$? Since $Abs_{ind(\mathcal{H})}(\mathcal{B})$ is an abstraction for $Quo_{ind(\mathcal{H})}(\mathcal{B})$, by Definition 10 it holds that $\mathcal{T}_{\mathcal{C}} \subseteq \mathcal{T}'_{\mathcal{C}}, \mathcal{I} \subseteq \mathcal{I}', \mathcal{U} \subseteq \mathcal{U}'$ and that $\mathcal{T}_{\mathcal{D}}(\mathrm{u}) \subseteq \mathcal{T}'_{\mathcal{D}}(\mathrm{u})$ for $\mathrm{u} \in ind(\mathrm{C})$. Note that in general most of the inclusions above are strict [4,14]. By the construction of $Abs_{\mathcal{H}}(\mathcal{B})$, it holds that $\mathcal{ST}_{\mathcal{C}} \subseteq \mathcal{ST}'_{\mathcal{C}}, \mathcal{I} \subseteq \mathcal{I}', \mathcal{U} \subseteq \mathcal{U}'$. The following lemma shows that it holds also $\mathcal{ST}_{\mathcal{D}} \subseteq \mathcal{ST}'_{\mathcal{D}}$:

Lemma 3. *Consider the abstraction $Abs_{\mathcal{H}}(\mathcal{B})$ as defined in Definition 11. Then, it holds that $\mathcal{ST}_{\mathcal{D}}(c) \subseteq \mathcal{ST}'_{\mathcal{D}}(c)$, for all $c \in C$.*

Proof sketch: Fix $\mathrm{c} \in \mathrm{C}$. Assume that $((m, B), \mu') \in \mathcal{ST}_{\mathcal{D}}(\mathrm{c})$. Then, by Definition 9, there exists $\boldsymbol{x} \in B$ and a transition $((m, \boldsymbol{x}), \mu) \in Steps_{\mathcal{D}}(\mathrm{c})$ such that $\mu' \in lift_{\mathcal{B}}(\mu)$. For $i = 1, \ldots, q_{\mathrm{c}}$, let $(m_i, \boldsymbol{x_i}) = update_i(m, \boldsymbol{x})$, and let (m_i, B_i) be the abstract states corresponding to the distribution μ' (cf. Definition 8), i.e., $\mu'(m_i, B_i) = \sum_{\{j | (m_j, B_j) = (m_i, B_i)\}} \mu(m_j, \boldsymbol{x_j})$. Obviously, $((m, \boldsymbol{x}), (m_i, \boldsymbol{x}_i)) \in T_{\mathcal{D}}(\mathrm{u}_i)$. Since $\boldsymbol{x_i} \in B_i$ it holds that $((m, B), (m_i, B_i)) \in \mathcal{T}_{\mathcal{D}}(\mathrm{u}_i) \subseteq \mathcal{T}'_{\mathcal{D}}(\mathrm{u}_i)$ for $i = 1, \ldots, q_{\mathrm{c}}$. By Definition 11, we have that $((m, B), \mu') \in \mathcal{ST}_{\mathcal{D}}(\mathrm{c})$. ■

The set of transitions $\mathcal{ST}'_{\mathcal{D}}(\mathrm{c})$ is indeed an over-approximation, which is illustrated as follows.

Example 2. Consider the fragment of the abstraction depicted in Fig. 3 in which we assume that the transitions correspond to the guarded command c with $q_{\mathrm{c}} = 4$: $condition \rightarrow p_1 : up_1 + \ldots + p_4 : up_4$. The abstract states are represented by circles, labelled with the corresponding tuple. The concrete states are represented by black points, labelled with only the evaluation of the variables (assume that all of them are different). Thus s_0 represents state (m_0, s_0) and so on. Arrows are transitions in the concrete models, where the labels represent the probability p_i of the corresponding update up_i of c. Assume that all of the concrete states are different and are not on borders. (Note: only parts of successor distributions are depicted, and we assume that other parts (e.g. for state (m_0, s_1)) lead to abstract states outside the depicted fragment.)

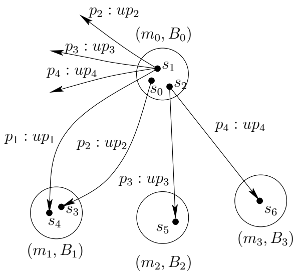

Fig. 3. Abstracting abstract discrete transitions

Now we consider the distribution $\mu^* \in Distr(\mathcal{B})$ which is defined as follows: $\mu^*(m_1, B_1) = p_1 + p_2$, $\mu^*(m_2, B_2) = p_3$ and $\mu^*(m_3, B_3) = p_4$. By the above assumption, no concrete successor distributions of s_0, s_1 or s_2 could induce μ^* according Definition 8. Thus, by Definition 9, $((m_0, B_0), \mu^*) \notin \mathcal{ST}_{\mathcal{D}}(\mathrm{c})$. On the other hand, it holds $((m_0, B_0), (m_1, B_1)) \in \mathcal{T}'_{\mathcal{D}}(up_i)$ for $i = 1, 2$, $((m_0, B_0), (m_2, B_2)) \in \mathcal{T}'_{\mathcal{D}}(up_3)$, and $((m_0, B_0), (m_3, B_3)) \in \mathcal{T}'_{\mathcal{D}}(up_4)$. Thus, by Definition 11 we have that $((m_0, B_0), \mu^*) \in \mathcal{ST}'_{\mathcal{D}}(\mathrm{c})$.

Lemma 3 implies that $Abs_{\mathcal{H}}(\mathcal{B})$ is an abstraction of $Quo_{\mathcal{H}}(\mathcal{B})$. Thus:

Theorem 1. *For every probabilistic hybrid automaton H, for every abstraction $Abs_{ind(\mathcal{H})}(\mathcal{B})$ of the induced hybrid automaton $ind(H)$, the safety of $Abs_{\mathcal{H}}(\mathcal{B})$ implies the safety of H.*

5 Experiments

We implemented our method in the prototypical tool `ProHVer` (probabilistic hybrid automata verifier). It combines a modified version of `PHAVer` [17] to obtain the abstract state space with a component to compute an upper probability bound for the reachability problem using value iteration in the induced abstract probabilistic automaton. To show the applicability of our approach, we applied `ProHVer` on several case studies, which are small but diverse in the nature of their behaviour. Even though in each of them we considered bounded reachability (by using a clock variable to bound the time) to get result other than 1, our method is not in principal restricted to time bounded reachability.

`PHAVer` covers the reachable continuous space (per discrete location) by polyhedra of a maximal width. It can split locations (introducing new discrete locations) if the over-approximations carried out while constructing this covering are too coarse. This is effective in practice. But if we attempt to improve precision by reducing the maximal width, the resulting covering and location splits can look entirely different. This carries over to the probabilistic side.

This phenomenon of `PHAVer` may induce situations, where that reduced width setting does not lead to tighter probability bounds. Usually it does.

We here consider the thermostat example depicted in Fig. 4, which is extended from the one in [14]. There are four modes: $Cool, Heat, Check$ and $Error$. The latter mode models the occurrence of a failure, where the temperature sensor gets stuck at the last checked temperature. The set of variables are $\{t, x, T\}$ where T represents the temperature, t represents a local timer and x is used to measure the total time passed so far. Thus, in all modes it holds that $\dot{x} = 1$ and $\dot{t} = 1$. In each mode there is also an invariant constraint restricting the set of state space for this mode. Invariant constraints are only for the sake of convenience and comparison with [14].

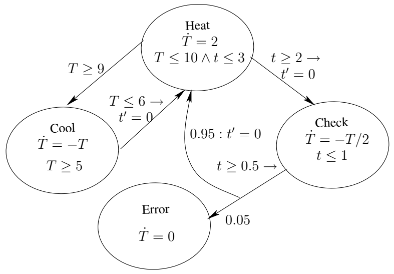

Fig. 4. A probabilistic hybrid automaton for the thermostat

The given initial condition is $m = Heat \wedge t = 0 \wedge x = 0 \wedge 9 \leq T \leq 10$. The unsafe constraint is $m = Error \wedge x \leq 5$, which corresponds to reaching the $Error$ mode within time 5. Assume that the probability threshold for this risk is specified to be 0.2. `ProHVer` can verify this nontrivial system and property, and will answer that the system is safe, the upper bound computed is 0.097.

In Fig. 5, we give probability bounds and performance statistics (time to build the abstraction – the value iteration time is negligible, and number of constructed abstract states) for different time bounds. For the left (right) part we instantiated the splitting interval for variable x with length 2 (respectively length 10). This governs the refinement technique of PHAVer. The time needed for the analysis as well as the number of states of the abstract transition systems grows about linearly in the time bound, though with oscillations. Comparing the left and the right side, we see that for the larger interval we need less resources, as was to be expected. Due to the way PHAVer splits locations along intervals, for some table entries, we see somewhat counter-intuitive behaviour. We observe that bounds do not necessarily improve with decreasing interval length. This is because PHAVer does not guarantee abstractions with smaller intervals to be an improvement, though they are in most cases. Furthermore, the abstraction we obtain from PHAVer can not guarantee probability bounds to increase monotonically with the time bound. This is because a slightly increased time bound might induce an entirely different abstraction, leading to a tighter probability bound, and thus giving the impression of a decrease in probability, even though the actual maximal probability indeed stays the same or increases.

time	interval length 2			interval length 10		
bound	prob.	build (s)	#states	prob.	build (s)	#states
2	0	0	11	0	0	8
4	0.05	0	43	1	0	12
5	0.097	1	58	1	0	13
20	0.370	20	916	1	1	95
40	0.642	68	2207	0.512	30	609
80	0.884	134	4916	1	96	1717
120	0.940	159	4704	0.878	52	1502
160	0.986	322	10195	0.954	307	4260
180	0.986	398	10760	0.961	226	3768
600	1.0	1938	47609	1	1101	12617

Fig. 5. Thermostat performance

In addition to the thermostat case, we have considered a selection of other case studies: a bouncing ball assembled from different materials, a water level control system where sensor values may be delayed probabilistically, and an autonomous lawn-mower that uses a probability bias to avoid patterns on lawns. As safety problems to be verified we considered (time bounded) reachability properties. We varied the time bounds and other parameters of the analysis, leading to different upper bounds of varying precision. Mostly, the upper bounds we could obtain were tight or exact (checked by manual inspection). Due to space restrictions, we have put the complete descriptions of all case studies and corresponding results on our preliminary homepage for the tool at:

http://depend.cs.uni-sb.de/tools/prohver

6 Conclusions

In this paper we have discussed how to check safety properties for probabilistic hybrid automata. These models and properties are of central importance for the design and verification of emerging wireless and embedded real-time applications. Moreover, being based on arbitrary abstractions computed by tools for the analysis of non-probabilistic hybrid automata, improvements in effectivity of such tools directly carry over to improvements in effectivity of the technique we describe. The applicability of our approach has been demonstrated on a number of case studies, tackled using a prototypical implementation.

As future work we are investigating whether our approach can be adapted to the safety verification problem for more general probabilistic hybrid systems [7,8], that is, systems with stochastic differential equations instead of ordinary differential equations.

Acknowledgement. The work of Lijun Zhang was partially supported in part by MT-LAB, a VKR Centre of Excellence. Part of this work was done while Lijun Zhang was at Saarland University and Computing Laboratory, Oxford University. The work of Zhikun She was partly supported by Beijing Nova Program, the one of Stefan Ratschan has been supported by GAČR grant 201/08/J020 and by the institutional research plan AV0Z100300504. Holger Hermanns and Ernst Moritz Hahn were supported by the NWO-DFG bilateral project ROCKS, by the DFG as part of the Transregional Collaborative Research Centre SFB/TR 14 AVACS, and by the European Community's Seventh Framework Programme under grant agreement n^o 214755. The work of Ernst Moritz Hahn has been also supported by the Graduiertenkolleg "Leistungsgarantien für Rechnersysteme".

References

1. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theoretical Computer Science 138, 3–34 (1995)
2. Preußig, J., Kowalewski, S., Wong-Toi, H., Henzinger, T.: An algorithm for the approximative analysis of rectangular automata. In: Ravn, A.P., Rischel, H. (eds.) FTRTFT 1998. LNCS, vol. 1486, p. 228. Springer, Heidelberg (1998)
3. Clarke, E., Fehnker, A., Han, Z., Krogh, B., Stursberg, O., Theobald, M.: Verification of hybrid systems based on counterexample-guided abstraction refinement. In: Garavel, H., Hatcliff, J. (eds.) TACAS 2003. LNCS, vol. 2619, pp. 192–207. Springer, Heidelberg (2003)
4. Ratschan, S., She, Z.: Safety verification of hybrid systems by constraint propagation based abstraction refinement. ACM Transactions on Embedded Computing Systems 6 (2007)
5. Altman, E., Gaitsgory, V.: Asymptotic optimization of a nonlinear hybrid system governed by a markov decision process. SIAM Journal of Control and Optimization 35, 2070–2085 (1997)
6. Sproston, J.: Decidable model checking of probabilistic hybrid automata. In: Joseph, M. (ed.) FTRTFT 2000. LNCS, vol. 1926, pp. 31–45. Springer, Heidelberg (2000)
7. Bujorianu, M.L.: Extended stochastic hybrid systems and their reachability problem. In: Alur, R., Pappas, G.J. (eds.) HSCC 2004. LNCS, vol. 2993, pp. 234–249. Springer, Heidelberg (2004)
8. Bujorianu, M.L., Lygeros, J., Bujorianu, M.C.: Bisimulation for general stochastic hybrid systems. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 198–214. Springer, Heidelberg (2005)
9. Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. Automatica 44, 2724–2734 (2008)
10. Blom, H., Lygeros, J.: Stochastic Hybrid Systems: Theory and Safety Critical Applications. Lecture Notes in Control and Information Sciences, vol. 337. Springer, Heidelberg (2006)
11. Fränzle, M., Hermanns, H., Teige, T.: Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 172–186. Springer, Heidelberg (2008)
12. Teige, T., Fränzle, M.: Constraint-based analysis of probabilistic hybrid systems. In: ADHS (2009)
13. Segala, R., Lynch, N.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2, 250–273 (1995)
14. Alur, R., Dang, T., Ivancic, F.: Predicate abstraction for reachability analysis of hybrid systems. ACM Transactions on Embedded Computing Systems 5, 152–199 (2006)

15. D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reachability analysis of probabilistic systems by successive refinements. In: de Luca, L., Gilmore, S. (eds.) PROBMIV 2001, PAPM-PROBMIV 2001, and PAPM 2001. LNCS, vol. 2165, pp. 39–56. Springer, Heidelberg (2001)
16. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: Gupta, A., Malik, S. (eds.) CAV 2008. LNCS, vol. 5123, pp. 162–175. Springer, Heidelberg (2008)
17. Frehse, G.: Phaver: Algorithmic verification of hybrid systems past hytech. In: Morari, M., Thiele, L. (eds.) HSCC 2005. LNCS, vol. 3414, pp. 258–273. Springer, Heidelberg (2005)
18. Kwiatkowska, M.Z., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. Theoretical Computer Science 282, 101–150 (2002)
19. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata. Journal of Computer and System Sciences 57, 94–124 (1998)
20. Lafferriere, G., Pappas, G.J., Yovine, S.: A new class of decidable hybrid systems. In: Vaandrager, F.W., van Schuppen, J.H. (eds.) HSCC 1999. LNCS, vol. 1569, pp. 137–151. Springer, Heidelberg (1999)
21. Kwiatkowska, M., Norman, G., Parker, D.: Stochastic games for verification of probabilistic timed automata. In: Ouaknine, J., Vaandrager, F.W. (eds.) FORMATS 2009. LNCS, vol. 5813, pp. 212–227. Springer, Heidelberg (2009)
22. Davis, M.: Markov Models and Optimization. Chapman & Hall, Boca Raton (1993)
23. Arnold, L.: Stochastic Differential Equations: Theory and Applications. Wiley - Interscience, Chichester (1974)
24. Hu, J., Lygeros, J., Sastry, S.: Towars a theory of stochastic hybrid systems. In: Lynch, N.A., Krogh, B.H. (eds.) HSCC 2000. LNCS, vol. 1790, pp. 160–173. Springer, Heidelberg (2000)
25. Bujorianu, M.L., Lygeros, J.: Toward a general theory of stochastic hybrid systems. In: Stochastic Hybrid Systems Theory and Safety Critical Applications, pp. 3–30 (2006)
26. Julius, A.A.: Approximate abstraction of stochastic hybrid automata. In: Hespanha, J.P., Tiwari, A. (eds.) HSCC 2006. LNCS, vol. 3927, pp. 318–332. Springer, Heidelberg (2006)
27. Bujorianu, M.L., Lygeros, J., Langerak, R.: Reachability analysis of stochastic hybrid systems by optimal control. In: Egerstedt, M., Mishra, B. (eds.) HSCC 2008. LNCS, vol. 4981, pp. 610–613. Springer, Heidelberg (2008)
28. Jonsson, B., Larsen, K.G.: Specification and refinement of probabilistic processes. In: LICS, pp. 266–277 (1991)