

Comparing Two Approaches to Include Stochasticity in Hybrid Automata^{*}

Lisa Willemsen¹[0000-0002-0418-9854], Anne Remke^{2,1}[0000-0002-5912-4767], and Erika Ábrahám³[0000-0002-5647-6134]

¹ University of Twente, Enschede, The Netherlands

`l.c.willemsen@utwente.nl`

² University of Münster, Münster, Germany

`anne.remke@uni-muenster.de`

³ RWTH Aachen University, Aachen, Germany

`abraham@cs.rwth-aachen.de`

Abstract. Different stochastic extensions of hybrid automata have been proposed in the past, with unclear expressivity relations between them. To structure and relate these modeling languages, in this paper we formalize two alternative approaches to extend hybrid automata with stochastic choices of discrete events and their time points. The first approach, which we call decomposed scheduling, adds stochasticity via stochastic races, choosing random time points for the possible discrete events and executing a winner with an earliest time. In contrast, composed scheduling first samples the time point of the next event and then the event to be executed at the sampled time point. We relate the two approaches regarding their expressivity and categorize available stochastic extensions of hybrid automata from the literature.

Keywords: Formal Modelling · Stochastic Hybrid Models · Classification · Expressivity.

1 Introduction

Hybrid automata (HA) [11] are well-suited to model the interplay of continuous and discrete behavior. Hybrid automata naturally exhibit non-determinism, e.g., *discrete non-determinism* via multiple simultaneously enabled discrete events (so-called jumps), or *time non-determinism* via time evolution with non-deterministic duration. In this paper we focus on these aspects and assume that the initial state, successor states of jumps, as well as the continuous evolution of the system state during time elapse are deterministic.

During the execution of a hybrid automaton, every non-deterministic choice has to be resolved by a scheduler. Hybrid automata have been extended with stochastic choices in multiple ways, leading to *stochastic hybrid models (SHM)* [3, 15] with different features and expressivity. In most existing formalisms on

^{*} This work is supported by the DFG grant 471367371.

SHM, all decisions are made randomly and they completely replace the non-determinism present in the underlying HA.

For example discrete probability distributions have been added to decidable subclasses of hybrid automata [21,22], which can be analyzed e.g. with the SISAT tool using abstraction [23]. Another possible extension are stochastic delays via stochastic resets or random clocks [6,16], which are sampled from a continuous distribution to determine how much time should pass between consecutive discrete jumps. In the PROHVER [7] framework, stochastic resets are abstracted to non-deterministic probabilistic resets. Another over-approximating approach [9] discretizes the support of the random variables and then abstracts to Markov decision processes. Some of these formalisms model stochastic choices by stochastic kernels [3,15], each of them responsible for random decisions of a certain kind; we refer to this technique as *composed scheduling*. Others use several stochastically independent random decisions that are in “race” [6,16]; we call this approach *decomposed scheduling*. Due to these differences and diverging mathematical notation, it is often hard to compare existing formalisms with respect to their expressivity.

Contribution. (i) In this paper, we formalize two stochastic HA extensions, that implement composed respectively decomposed scheduling. (ii) We define the stochastic processes induced by each of the approaches. (iii) We relate the expressivity of the two approaches and show that composed scheduling is more expressive than decomposed scheduling. (iv) We discuss how existing formalisms implement such stochastic choices and relate different lines of work.

Outline. Section 2 introduces HA and the necessary stochastic notation. Section 3 formalizes and relates the two HA extensions. Section 4 discusses related work and classifies existing formalisms. Section 5 concludes the paper.

2 Fundamentals

Let \mathbb{R} denote the set of all real numbers, $\mathbb{R}_{\geq 0}$ the nonnegative reals, \mathbb{N} the natural numbers (including zero) and \mathbb{Z} the integers. For a set S , 2^S is the set of all subsets of S . We start with introducing hybrid automata in Section 2.1 and recall some basic definitions from probability theory in Section 2.2.

2.1 Hybrid Automata

Hybrid automata extend discrete transition systems with the notion of time and continuous evolution. In the below standard definition [10] we omit modeling constructs for parallel composition, as they are not central for this work.

Definition 1 (Hybrid automata: Syntax). A hybrid automaton (HA) is a tuple $\mathcal{H} = (Loc, Var, Flow, Inv, Edge, Init)$ with the following components:

- *Loc* is a non-empty finite set of locations or control modes.

- $Var = \{x_1, \dots, x_d\}$ is a finite ordered set of variables. We call d the dimension, $\nu \in \mathbb{R}^d$ a valuation, and $\sigma = (\ell, \nu) \in Loc \times \mathbb{R}^d = \Sigma$ a state of \mathcal{H} .
- $Flow : Loc \rightarrow (\mathbb{R}^d \rightarrow \mathbb{R}^d)$ specifies for each location its flow or dynamics.
- $Inv : Loc \rightarrow 2^{\mathbb{R}^d}$ specifies an invariant for each location. We define $\Sigma_{Inv} = \{(\ell, \nu) \in \Sigma \mid \nu \in Inv(\ell)\}$.
- $Edge \subseteq Loc \times 2^{\mathbb{R}^d} \times (\mathbb{R}^d \rightarrow \mathbb{R}^d) \times Loc$ is a finite set of discrete transitions or jumps. For a jump $(\ell_1, g, r, \ell_2) \in Edge$, ℓ_1 and ℓ_2 are its source resp. target locations, g its guard, and r its reset.
- $Init : Loc \rightarrow 2^{\mathbb{R}^d}$ defines initial valuations. We call a state $(\ell, \nu) \in \Sigma$ initial if $\nu \in Inv(\ell) \cap Init(\ell)$.

Executions of a hybrid automaton evolve an initial state by time steps and discrete steps. *Time steps (flows)* model continuous evolution of the variable values according to the flow condition of the current location, while satisfying the current location's invariant. When flows define constant derivatives for all variables then we talk about *linear behavior*, for linear predicates (i.e., linear differential equations) about *linear dynamics*, and in the case of more expressive predicates (involving e.g. polynomials or trigonometric functions) about *nonlinear dynamics*. *Discrete steps (jumps)* $(\ell, g, r, \ell') \in Edge$ can move the control from location ℓ to ℓ' and change the valuation from $\nu \in \mathbb{R}^d$ to $r(\nu) \in \mathbb{R}^d$, assuming that the jump is *enabled* in (ℓ, ν) , i.e. $\nu \in g$ and $r(\nu) \in Inv(\ell')$.

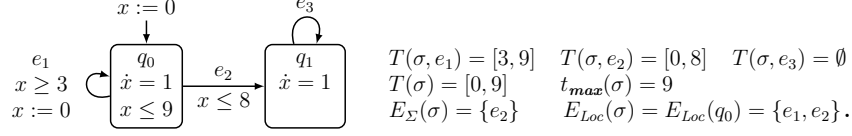
Definition 2 (Hybrid automata: Semantics). For a hybrid automaton $\mathcal{H} = (Loc, Var, Flow, Inv, Edge, Init)$ of dimension d , its operational semantics is given by the following rules:

$$\frac{\begin{array}{l} \ell \in Loc \quad \nu, \nu' \in \mathbb{R}^d \quad t \in \mathbb{R}_{\geq 0} \quad f : [0, t] \rightarrow \mathbb{R}^d \quad df/dt = \dot{f} : (0, t) \rightarrow \mathbb{R}^d \\ f(0) = \nu \quad f(t) = \nu' \quad \forall t' \in (0, t). \dot{f}(t') = Flow(\ell)(f(t')) \\ \forall t' \in [0, t]. f(t') \in Inv(\ell) \end{array}}{(\ell, \nu) \xrightarrow{t} (\ell, \nu')} \quad \text{Flow}$$

$$\frac{e = (\ell, g, r, \ell') \in Edge \quad \nu, \nu' \in \mathbb{R}^d \quad \nu \in g \quad \nu' = r(\nu) \quad \nu' \in Inv(\ell')}{(\ell, \nu) \xrightarrow{e} (\ell', \nu')} \quad \text{Jump}$$

Let $\mathcal{H} = (Loc, Var, Flow, Inv, Edge, Init)$ be a HA of dimension d . A *path* of \mathcal{H} is a finite or infinite sequence $\pi = \sigma_0 \xrightarrow{t_0} \sigma'_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{t_1} \dots$ of alternating time steps and jumps with $\sigma_0 \in \Sigma_{Inv}$; π is said to be *initial* if σ_0 is initial, we define its *length* $len(\pi)$ as the number of jumps in it, and its *duration* $dur(\pi)$ as the sum of the durations of all of its time steps. Let $\Pi(\sigma)$ and $\Pi_{fn}(\sigma)$ be the set of all infinite resp. finite paths starting in $\sigma \in \Sigma_{Inv}$. A state $\sigma \in \Sigma$ of \mathcal{H} is *reachable* iff there is an initial path leading to it.

An infinite path is *time-convergent* if its duration is finite, and *time-divergent* otherwise. A state of \mathcal{H} has a *timelock* iff all infinite paths starting in it are time-convergent; \mathcal{H} has a *timelock* if any of its reachable states has a timelock, and is *timelock-free* otherwise. An infinite path is *Zeno* iff it is time-convergent and contains infinitely many jumps. \mathcal{H} is *Zeno-free* if it has no initial Zeno path.

Fig. 1: A hybrid automaton and characteristics of its state $\sigma = (q_0, \nu)$, $\nu(x) = 0$.

We discuss how choices over jumps and the length of time steps are made stochastically in Section 3, where we use the following notions, partly generalized to hybrid automata from [3] and illustrated in Figure 1. For $e \in Edge$ and $\sigma \in \Sigma_{Inv}$, we define

$$\begin{aligned} T(\sigma, e) &= \{t \in \mathbb{R}_{\geq 0} \mid \exists \sigma', \sigma'' \in \Sigma_{Inv}. \sigma \xrightarrow{t} \sigma' \wedge \sigma' \xrightarrow{e} \sigma''\}, & T(\sigma) &= \bigcup_{e \in Edge} T(\sigma, e), \\ t_{max}(\sigma) &= \sup\{t \in \mathbb{R} \mid \exists \sigma' \in \Sigma_{Inv}. \sigma \xrightarrow{t} \sigma'\}, \\ E_{\Sigma}(\sigma) &= \{e \in Edge \mid \exists \sigma' \in \Sigma_{Inv}. \sigma \xrightarrow{e} \sigma'\}, \\ E_{Loc}(\ell) &= \bigcup_{\nu \in \mathcal{V}} E_{\Sigma}((\ell, \nu)), & E_{Loc}((\ell, \nu)) &= E_{Loc}(\ell). \end{aligned}$$

We call σ *jump-enabled*⁴ iff $T(\sigma) \neq \emptyset$, and *immediate-jump-enabled* iff $E_{\Sigma}(\sigma) \neq \emptyset$.

2.2 Basic Stochastic Notions

A random *experiment* has an uncertain outcome from a *sample space* Ω , whose subsets are called *events*. A σ -*algebra* \mathcal{F} is a set of events containing the maximal event Ω and being closed under complement and countable union. The standard Borel σ -algebra $\mathcal{B}(\Omega)$ is the smallest σ -algebra containing all open events. An event is \mathcal{F} -*measurable* if it is in \mathcal{F} . The pair (Ω, \mathcal{F}) is called a *measurable space*.

Given (Ω, \mathcal{F}) , a *probability measure* is a function $Pr : \mathcal{F} \rightarrow [0, 1] \subseteq \mathbb{R}$ with (i) $Pr(\Omega) = 1$, (ii) $Pr(E) = 1 - Pr(\bar{E})$ for all $E \in \mathcal{F}$ and (iii) $Pr(\bigcup_{i=0}^{\infty} E_i) = \sum_{i=0}^{\infty} Pr(E_i)$ for any $E_i \in \mathcal{F}$ with $E_i \cap E_j = \emptyset$ for all $i, j \in \mathbb{N}$, $i \neq j$.

A *probability space* is a triple $(\Omega, \mathcal{F}, Pr)$ with (Ω, \mathcal{F}) a measurable space and Pr a probability measure for (Ω, \mathcal{F}) .

Let (Ω, \mathcal{F}) and (S, Σ) be measurable spaces, $X : \Omega \rightarrow S$, $s \in S$ and $\sigma \in \Sigma$. We define $X \sim s$ to be $\{\omega \in \Omega \mid X(\omega) \sim s\}$ and $X^{-1}(\sigma) = \bigcup_{s \in \sigma} (X \sim s)$ with $\sim \in \{\leq, <, =, >, \geq\}$. X is *measurable* (wrt. (Ω, \mathcal{F}) and (S, Σ)) if $X^{-1}(\sigma) \in \mathcal{F}$ for all $\sigma \in \Sigma$. A *random variable* is a measurable function $X : \Omega \rightarrow S$; we call $X(\omega)$ the *realization* of X for $\omega \in \Omega$.

For the following we instantiate $\Omega = S = \mathbb{R}_{\geq 0}$, $\mathcal{F} = 2^{\mathbb{R}_{\geq 0}}$ and X the identity. For $f : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ we define its *support* as $supp(f) = \{\omega \in \mathbb{R} \mid f(\omega) > 0\}$, with:

- If $supp(f)$ is countable and $\sum_{\omega \in supp(f)} f(\omega) = 1$, f is called a *discrete probability distribution*, which induces the unique probability measure $Pr : 2^{\mathbb{R}_{\geq 0}} \rightarrow [0, 1]$ with $Pr(E) = \sum_{\omega \in E \cap supp(f)} f(\omega)$ for all $E \subseteq \mathbb{R}_{\geq 0}$.

⁴ In the original definition, this is called *non-blocking*.

- If f is absolute continuous with $\int_0^\infty f(\omega)d\omega = 1$ then f is called a *continuous probability distribution* or a *probability density function (PDF)*, which induces for all $a \in \mathbb{R}_{\geq 0}$ the unique probability measure $Pr : 2^{\mathbb{R}_{\geq 0}} \rightarrow [0, 1]$ with $Pr(X \leq a) = \int_0^a f(\omega)d\omega$, and the *cumulative distribution function (CDF)* $F : \mathbb{R}_{\geq 0} \rightarrow [0, 1]$ with $F(a) = Pr(X \leq a)$.

We denote the set of all discrete resp. continuous probability distributions by $Dist_d$ resp. $Dist_c$ and call elements from $Dist = Dist_d \cup Dist_c$ *probability distributions*. A random variable is *discrete* if its underlying probability measure Pr is induced by a discrete probability distribution, and *continuous* otherwise. By \mathbb{X}_d and \mathbb{X}_c we denote the set of all discrete resp. continuous random variables.

Given two measurable spaces $(\Omega_1, \mathcal{F}_1)$ and $(\Omega_2, \mathcal{F}_2)$, a *stochastic kernel from $(\Omega_1, \mathcal{F}_1)$ to $(\Omega_2, \mathcal{F}_2)$* [14] is a function $\kappa : \mathcal{F}_2 \times \Omega_1 \rightarrow [0, 1]$ with:

- For each $E_2 \in \mathcal{F}_2$, the function $f_{E_2}^\kappa : \Omega_1 \rightarrow [0, 1]$ with $f_{E_2}^\kappa(\omega_1) = \kappa(E_2, \omega_1)$ is measurable w.r.t. $(\Omega_1, \mathcal{F}_1)$ and $([0, 1], \mathcal{B}([0, 1]))$.
- For each $\omega_1 \in \Omega_1$, the function $Pr_{\omega_1}^\kappa : \mathcal{F}_2 \rightarrow [0, 1]$ with $Pr_{\omega_1}^\kappa(E_2) = \kappa(E_2, \omega_1)$ is a probability measure on $(\Omega_2, \mathcal{F}_2)$.

Stochastic kernels are used to express the state-dependent probability $\kappa(E_2, \omega_1)$ of event $E_2 \in \mathcal{F}_2$ in system state $\omega_1 \in \Omega_1$. κ is *discrete* if each $Pr_{\omega_1}^\kappa$ can be induced by a discrete probability distribution, and *continuous* otherwise.

A *stochastic process* over an index set T is a family of random variables $\{X(t) \mid t \in T\}$, defined over a common probability space and taking values in the same measurable space. In this work we use continuous-time stochastic processes with $T \subseteq \mathbb{R}_{\geq 0}$. A stochastic process has the *Markov property* if its future is independent of its past evolution [14].

3 Extending Hybrid Automata with Stochasticity

In this section we formalize two stochastic HA extensions: *Composed scheduling*, introduced in Section 3.1, randomly chooses first a delay and then a jump to be taken after the delay. Conversely, *decomposed scheduling*, introduced in Section 3.2, chooses a delay for each jump separately, where the jump with the minimal delay is taken. Our approach chooses delays from $\mathbb{R}_{\geq 0}$, which might be unrealizable due to invariants, or after which no jump might be enabled, so we need to introduce mechanisms to manage these cases. We mention that other formalisms like [3] assume an “oracle” and can therefore sample only over realizable time delays after which there is an enabled jump.

To put a clear focus on the differences between composed and decomposed scheduling, our HA definition assumes deterministic flows and jump resets. In addition, for simplicity we assume in the following a unique initial state. However, our languages and results can easily be extended to relax these restrictions.

In Sections 3.1 and 3.2 we assume that all invariants evaluate to true; we use \top to denote the trivial invariant, i.e. $\top(\ell) = \mathbb{R}^d$ for all $\ell \in Loc$. In Section 3.5 we discuss which adaptations in the definitions are required to apply (de-)composed scheduling to HA with invariants.

3.1 Composed Scheduling

In *composed scheduling*, the durations of time steps and the jumps to be taken are sampled according to two stochastic kernels Ψ_c resp. Ψ_d . For a state σ , the probability distributions induced by Ψ_c and Ψ_d are denoted $\text{Dist}_\sigma^{\Psi_c}$ resp. $\text{Dist}_\sigma^{\Psi_d}$.

To schedule time delays, approaches like e.g. [3] sample only durations after which there exists an enabled jump. This alternative is meaningful for decidable subclasses only (as one needs to determine valid time durations) and for them it would result in the same expressivity as our approach, which is as follows.

Assume a d -dimensional hybrid automaton $\mathcal{H}_{in} = (Loc, Var, Edge_{in}, Act, \top, Init)$. Without an oracle, it might happen that Ψ_c samples a time duration after which there are no enabled jumps. To handle such cases, we introduce for each location $\ell \in Loc$ a unique (*composed*) *resampling jump* $\epsilon_\ell^r = (\ell, g, r, \ell)$ with guard $g = \mathbb{R} \setminus (\cup_{(\ell, g', r', \ell') \in Edge_{in}} g')$ and reset $r(\nu) = \nu$ for all $\nu \in \mathbb{R}^d$. Let $Edge_r^{comp} = \{\epsilon_\ell^r \mid \ell \in Loc\}$. We call $\mathcal{H} = (Loc, Var, Edge_{in} \cup Edge_r^{comp}, Act, \top, Init)$ the *composed resampling extension* of \mathcal{H}_{in} .

Definition 3 (Composed Syntax). A hybrid automaton with composed scheduling is a tuple $\mathcal{C} = (\mathcal{H}, \Psi_c, \Psi_d)$, where:

- $\mathcal{H} = (Loc, Var, Edge, Act, \top, Init)$ with states Σ is the composed resampling extension of a HA \mathcal{H}_{in} with trivial invariants and deterministic initial state.
- $\Psi_c : \mathcal{B}(\mathbb{R}_{\geq 0}) \times \Sigma \rightarrow [0, 1]$ is a continuous stochastic kernel from $(\Sigma, \mathcal{B}(\Sigma))$ to $(\mathbb{R}_{\geq 0}, \mathcal{B}(\mathbb{R}_{\geq 0}))$.
- $\Psi_d : \mathcal{B}(Edge) \times \Sigma \rightarrow [0, 1]$ is a discrete stochastic kernel from $(\Sigma, \mathcal{B}(\Sigma))$ to $(Edge, \mathcal{B}(Edge))$ such that $\text{supp}(\text{Dist}_\sigma^{\Psi_d}) \subseteq E_\Sigma(\sigma)$ for all $\sigma \in \Sigma$.

After each time step, if any non-resampling jump of \mathcal{H}_{in} is enabled then resampling is scheduled with probability 0, and otherwise with probability 1. In each jump successor state σ , a fresh delay is sampled according to $\text{Dist}_\sigma^{\Psi_c}$.

Definition 4 (Composed Semantics). Assume a HA with composed scheduling $\mathcal{C} = (\mathcal{H}, \Psi_c, \Psi_d)$. A path of \mathcal{C} is a path $\pi = \sigma_0 \xrightarrow{t_0} \sigma'_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{t_1} \dots$ of \mathcal{H} with $t_j \in \text{supp}(\text{Dist}_{\sigma'_j}^{\Psi_c})$ and $e_j \in \text{supp}(\text{Dist}_{\sigma'_j}^{\Psi_d})$ for all $j \geq 0$.

3.2 Decomposed Scheduling

Let $\mathcal{H}_{in} = (Loc, Var, Edge_{in}, Act, \top, Init)$ be a d -dimensional HA. Instead of centralized decisions via stochastic kernels, *decomposed scheduling* chooses jumps and the length of time steps by associating each jump with a continuous random variable from a non-empty set $\mathbb{X} = \{X_1, \dots, X_k\}$ via a function $a_{in} : Edge_{in} \rightarrow \{1, \dots, k\}$, such that two jumps with the same random variable are never enabled simultaneously. We call such a pair $(\mathcal{H}_{in}, a_{in})$ an \mathbb{X} -labeled HA.

The realisations of the random variables indicate the delay after which a jump with the given random variable should be taken; if no such jump is enabled then we again need a mechanism for resampling.

For each location $\ell \in Loc$ and random variable $X_i \in \mathbb{X}$ we introduce a (decomposed) resampling jump $\epsilon_{\ell,i}^r = (\ell, g_{\ell,i}, r, \ell)$ with reset $r(\nu) = \nu$ for all $\nu \in \mathbb{R}^d$ and guard $g_{\ell,i} = \mathbb{R}^d \setminus (\cup_{e \in \{e' = (\ell, g, r, \ell') \in Edge_{in} \mid a_{in}(e') = i\}} g)$. We extend the edge set to $Edge = Edge_{in} \cup Edge_r^{decomp}$ with the resampling edges $Edge_r^{decomp} = \{\epsilon_{\ell,i}^r \mid \ell \in Loc \wedge 1 \leq i \leq k\}$. Let $\mathcal{H} = (Loc, Var, Edge, Act, \top, Init)$.

We extend also a_{in} to cover the resampling edges, defining $a : Edge \rightarrow \{1, \dots, k\}$ with $a(e) = a_{in}(e)$ for $e \in Edge_{in}$ and $a(\epsilon_{\ell,i}^r) = i$ for $\epsilon_{\ell,i}^r \in Edge_r^{decomp}$. Let $a^{-1} : \{1, \dots, k\} \rightarrow 2^{Edge}$ with $a^{-1}(i) = \{e \in Edge \mid a(e) = i\}$ for $i = 1, \dots, k$.

We call (\mathcal{H}, a) the *decomposed resampling extension* of the \mathbb{X} -labeled HA $(\mathcal{H}_{in}, a_{in})$. Note that (\mathcal{H}, a) itself is an \mathbb{X} -labeled HA.

Definition 5 (Decomposed Syntax). A hybrid automaton with decomposed scheduling is a tuple $\mathcal{D} = (\mathcal{H}, \mathbb{X}, a)$ where:

- $\mathbb{X} = \{X_1, \dots, X_k\}$ is a finite non-empty ordered set of continuous random variables.
- (\mathcal{H}, a) with $\mathcal{H} = (Loc, Var, Edge, Act, \top, Init)$ is the decomposed resampling extension of some \mathbb{X} -labeled HA $(\mathcal{H}_{in}, a_{in})$, where \mathcal{H}_{in} has trivial invariants and a deterministic initial state.

We store the current realizations of the random variables in a sequence $\mathcal{R} = (x_1, \dots, x_k) \in \mathbb{R}_{\geq 0}^k$, and use $\mathcal{R}[j]$ to refer to x_j . The stochastic race between the random variables is “won” by the random variable which *expires* first as it has a smallest current realisation. The presence of resampling jumps ensures, that a jump can be scheduled, even if there is no enabled edge associated with the winning random variable. Note that, since all random variables follow a continuous probability distribution, the probability that two random variables expire at the same time is 0 and in this case it is irrelevant which jump is taken.

Definition 6 (Decomposed Semantics). Assume a HA with decomposed scheduling $\mathcal{D} = (\mathcal{H}, \mathbb{X}, a)$ with $\mathbb{X} = \{X_1, \dots, X_k\}$. A path of \mathcal{D} has the form $\pi = (\sigma_0, \mathcal{R}_0) \xrightarrow{t_0} (\sigma'_0, \mathcal{R}'_0) \xrightarrow{e_0} (\sigma_1, \mathcal{R}_1) \xrightarrow{t_1} \dots$ with $\sigma_i = (\ell_i, \nu_i)$, $\sigma'_i = (\ell_i, \nu'_i)$ such that $\sigma_0 \xrightarrow{t_0} \sigma'_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{t_1} \dots$ is a path of \mathcal{H} and such that for all $i \in \mathbb{N}$:

- $\mathcal{R}_i, \mathcal{R}'_i \in \mathbb{R}_{\geq 0}^k$ and for all $j \in \{1, \dots, k\}$, $\mathcal{R}_0[j]$ is sampled according to X_j 's probability distribution in σ_0 .
- $t_i = \min_{j \in \{1, \dots, k\}} \mathcal{R}_i[j]$ and $\mathcal{R}'_i[j] = \mathcal{R}_i[j] - t_i$ for all $j \in \{1, \dots, k\}$.
- $\mathcal{R}'_i[m_i] = 0$ for $m_i = a(e_i)$.
- The value $\mathcal{R}_{i+1}[m_i]$ is sampled according to X_{m_i} 's probability distribution in σ_{i+1} , and $\mathcal{R}_{i+1}[j] = \mathcal{R}'_i[j]$ for all $j \in \{1, \dots, k\} \setminus \{m_i\}$.

Example 1 ((De-)composed scheduling). We illustrate the application of composed and decomposed scheduling on the example HA depicted in Figure 2(a). Note that the resulting HA with (de-)composed scheduling in this example have different underlying stochastic processes (c.f. Section 3.3).

In the composed scheduling in Figure 2(b), choices over the time steps' durations are governed by a kernel Ψ_c , which characterises for each state with location

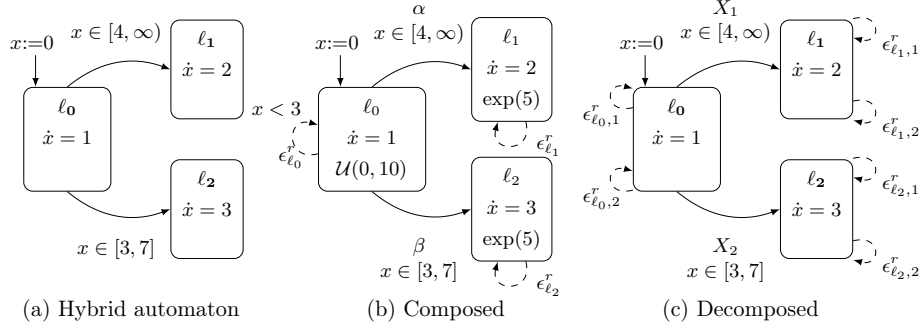


Fig. 2: Illustration to Example 1. Hybrid automaton shown in (a) is extended with either composed scheduling in (b) or decomposed scheduling in (c).

ℓ_0 a uniform distribution $\text{Dist}_{(\ell_0, \nu)}^{\Psi_c} = \mathcal{U}(0, 10)$, and an exponential distribution $\text{Dist}_{(\ell_i, \nu)}^{\Psi_c} = \exp(5)$ otherwise. The kernel Ψ_d characterises a discrete probability distribution over enabled jumps. Hence, $\Psi_d(e_1, (\ell_0, \nu)) = \alpha$, $\Psi_d(e_2, (\ell_0, \nu)) = \beta$ and $\Psi_d(\epsilon_{\ell_0}^r, (\ell_0, \nu)) = 1 - \alpha - \beta$, where e_1 denotes the jump to location ℓ_1 and e_2 the jump to location ℓ_2 . The values of α and β depend on the enabling status of e_1 and e_2 . Hence, $\alpha = 1$ and $\beta = 0$, if e_1 is the only jump enabled. Similarly, $\alpha = 0$, $\beta = 1$ in case e_2 is enabled but e_1 is not. If both jumps are enabled then we set $\alpha = 0.7$ and $\beta = 0.3$. If neither e_1 nor e_2 is enabled, then $\alpha = \beta = 0$ and the resampling edge is scheduled with probability 1.

In the decomposed scheduling in Figure 2(c), there is a race between two competing random variables X_1 and X_2 , and the "winner" decides on the delay as well as on the scheduled jump. For our example, in the initial state $\sigma_0 = (\ell_0, \nu)$ with $\nu(x) = 0$ we sample the values x_1 and x_2 for X_1 resp. X_2 from the exponential distribution with parameter 0.2. After a delay of $t = \min\{x_1, x_2\}$ a jump takes place. If $t = x_1$ then it is the unique enabled jump associated with X_1 (i.e. either the jump to ℓ_1 or the resampling jump $\epsilon_{\ell_0,1}^r$), otherwise if $t = x_2$ then it is the unique enabled jump associated with X_2 (i.e. either the jump to ℓ_2 or the resampling jump $\epsilon_{\ell_0,2}^r$).

3.3 Induced Stochastic Process

The semantics of a hybrid automaton with unique initial state and composed or decomposed scheduling is fully stochastic. The execution corresponds to a continuous-time stochastic process $\{X(t) \mid t \in \mathbb{R}_{\geq 0}\}$, where the random variable $X(t)$ takes values from the measurable space $(\Sigma, \mathcal{B}(\Sigma))$.

The corresponding probability space for finite paths of length $\text{len}(\pi) = n$ is given by $(\Omega, \mathcal{F}, Pr)$, where $\Omega = (\Sigma)^n$ and $\mathcal{F} = \mathcal{B}(\Omega)$. The probability measure Pr also depends on the chosen scheduling method. Since the probability of a single path is in most cases zero, we define measurable probabilities for *traces*.

Definition 7 (Trace). A trace of a HA \mathcal{H} is a finite sequence $\tau = (\sigma_0, e_0, e_1, \dots, e_n)$ composed of a state $\sigma_0 \in \Sigma_{Inv}$ and a (possibly empty) sequence of jumps e_0, \dots, e_n of \mathcal{H} . The trace τ is initial if σ_0 is an initial state of \mathcal{H} . A sub-trace of τ is a trace $\tau' = (\sigma_i, e_i, \dots, e_n)$ for some $1 \leq i \leq n$, where σ_i is reachable in \mathcal{H} through a path $\sigma_0 \xrightarrow{t_0} \sigma'_0 \xrightarrow{e_0} \sigma_1 \xrightarrow{t_1} \dots \xrightarrow{e_{i-1}} \sigma_i$. A trace of a HA with (de-)composed scheduling is a trace of its underlying HA.

The probability measure of a trace is obtained recursively by integrating over the enabling time of the first jump and taking into account the corresponding jump probabilities. Note that traces of \mathcal{C} might include resampling jumps.

Definition 8 (Composed Probability Measure). For a HA with composed scheduling $\mathcal{C} = (\mathcal{H}, \Psi_c, \Psi_d)$ and a trace $\tau = (\sigma_0, e_0, e_1, \dots, e_n)$ of \mathcal{C} we define $Pr_{\mathcal{C}}(\tau)$ to be 1 if $\tau = (\sigma_0)$, and otherwise

$$Pr_{\mathcal{C}}(\tau) = \int_{t \in T(\sigma_0, e_0)} \Psi_c(\sigma_0)(t) \cdot \Psi_d(\sigma'_0)(e_0) \cdot Pr_{\mathcal{C}}(\sigma_1, e_1, \dots, e_n) dt \quad (1)$$

where σ'_0 and σ_1 are the unique states of \mathcal{H} with $\sigma_0 \xrightarrow{t} \sigma'_0$ and $\sigma'_0 \xrightarrow{e_0} \sigma_1$.

Definition 9 (Decomposed Probability Measure). Assume a HA with decomposed scheduling $\mathcal{D} = (\mathcal{H}, \mathbb{X}, a)$, $\mathbb{X} = \{X_1, X_2, \dots, X_k\}$. For any state σ of \mathcal{H} and any $t \in \mathbb{R}_{\geq 0}$, let σ^t and be the unique state of \mathcal{H} with $\sigma \xrightarrow{t} \sigma^t$, and for any $e \in E_{\Sigma}(\sigma)$ let σ^e and be the unique state of \mathcal{H} with $\sigma \xrightarrow{e} \sigma^e$. For a trace $\tau = (\sigma_0, e_0, e_1, \dots, e_n)$ of \mathcal{D} we define

$$Pr_{\mathcal{D}}(\tau) = \int_0^{\infty} Dist_{X_1}(\sigma_0, t_1) \dots \int_0^{\infty} Dist_{X_k}(\sigma_0, t_k) P(\sigma_0^{\delta_0}, \mathcal{R}, e_0, e_1, \dots, e_n) dt_k \dots dt_1 \quad (2)$$

where $\delta_0 = \min\{t_m \mid 1 \leq m \leq k\}$, $\mathcal{R} \in \mathbb{R}_{\geq 0}^k$ with $\mathcal{R}[m] = t_m - \delta_0$ for all $1 \leq m \leq k$, and where P is defined as follows.

For any a trace $\tau = (\sigma_0, e_0, e_1, \dots, e_n)$ of \mathcal{D} and any $\mathcal{R} \in \mathbb{R}_{\geq 0}^k$ with $\sigma_0 = (\ell_0, \nu_0)$ and $m_0 = a(e_0)$ we set $P(\sigma_0, \mathcal{R}) = 1$ and for a non-empty sequence e_0, \dots, e_n :

$$P(\sigma_0, \mathcal{R}, e_0, \dots, e_n) = \begin{cases} 0 & \text{if } e_0 \notin E_{\Sigma}(\sigma_0) \text{ or } \mathcal{R}[m_0] \neq 0 \\ \int_0^{\infty} Dist_{X_{m_0}}(\sigma_0^{e_0}, t_{m_0}) \cdot P((\sigma_0^{e_0})^{\delta_1}, \mathcal{R}', e_1, \dots, e_n) dt_{m_0} & \text{otherwise,} \end{cases} \quad (3)$$

where $\delta_1 = \min(\{t_{m_0}\} \cup \{\mathcal{R}[m] \mid 1 \leq m \leq k \wedge m \neq m_0\})$, $\mathcal{R}'[m_0] = t_{m_0} - \delta_1$ and $\mathcal{R}'[m] = \mathcal{R}[m] - \delta_1$ for all $m \in \{1, \dots, k\} \setminus \{m_0\}$.

When decomposed scheduling is applied, the above-defined probability measure over traces (which might also include resampling jumps) is obtained by sampling all random variables from their corresponding probability distributions. Afterwards, it recursively computes the probability measure P of the trace for the sampled durations, after letting time elapse by the minimum realisation δ_0

under all random variables. In the definition of P in Equation 3, the first case applies if the jump e_0 is either disabled in the trace’s starting state or the realisation of its random variable is not yet expired (i.e. not 0); in this case, the probability of the trace is 0. Otherwise, we take the jump e_0 with successor state $\sigma_0^{e_0}$, resample the random variable X_{m_0} of e_0 , let again time elapse with the minimum realisation δ_1 over all random variables, and apply the definition recursively. Note that e_0 might be a resampling jump and that though several realisations can expire simultaneously, it might happen only with probability 0.

Remark 1. Hybrid automata with composed scheduling \mathcal{C} and HA with decomposed scheduling \mathcal{D} both extend the jump set of their underlying hybrid automaton \mathcal{H} . Therefore, \mathcal{C} and \mathcal{D} have more paths than \mathcal{H} . This means that Zeno paths are inherited from \mathcal{H} to \mathcal{C} and \mathcal{D} , however, Zeno-freedom of \mathcal{H} does not imply Zeno-freedom of \mathcal{D} and \mathcal{C} .

Consider for example the hybrid automaton \mathcal{H} from Figure 2(a) and its composed scheduling extension \mathcal{C} from Figure 2(b). The automaton \mathcal{H} is Zeno-free but \mathcal{C} does have Zeno paths, e.g. all paths that take the resampling jump ϵ_{ℓ_0} of ℓ_0 infinitely often. Even though the stochastic kernel of \mathcal{C} almost surely excludes such paths (i.e. if τ_k is the trace with k repeated ϵ_{ℓ_0} -jumps from the initial state σ_0 then $\lim_{k \rightarrow \infty} P(\tau_k) = 0$), changing the distribution in ℓ_0 from $\mathcal{U}(0, 10)$ to $\mathcal{U}(0, \frac{1-x}{2})$ would increase their probability to 1. Hence, modelers should carefully choose stochastic distributions in order to ensure that additional Zeno behavior is almost surely excluded.

3.4 Relation of Composed and Decomposed Scheduling

Previously, we discussed two different approaches on how hybrid automata can be extended with stochasticity. In this section we show that composed scheduling is *more expressive* than decomposed scheduling w.r.t. *Pr-equivalence*.

Definition 10 (Trace Probability Equivalence). *Let \mathcal{D} be a HA with decomposed scheduling, \mathcal{C} a HA with composed scheduling, and τ a common trace of \mathcal{D} and \mathcal{C} . The trace τ is Pr-equivalent in \mathcal{D} and \mathcal{C} iff $Pr_{\mathcal{D}}(\tau) = Pr_{\mathcal{C}}(\tau)$ and for each σ being either the first state of τ or the first state of a sub-trace of τ if holds for all $t \in \mathbb{R}_{\geq 0}$ that $Pr_{\mathcal{D}}^{\sigma}(X \leq t) = Pr_{\mathcal{C}}^{\sigma}(X \leq t)$, where X models the duration of a time step starting in σ .*

\mathcal{D} and \mathcal{C} are Pr-equivalent if the sets of their initial traces are equal and each of their initial traces is Pr-equivalent.

Theorem 1 (Expressivity Composed vs Decomposed Scheduling).

1. *Let \mathcal{D} be a HA with decomposed scheduling. Then there is a HA with composed scheduling \mathcal{C} such that \mathcal{D} and \mathcal{C} are Pr-equivalent.*
2. *There exists a HA with composed scheduling \mathcal{C}' such that there is no HA with decomposed scheduling \mathcal{D}' such that \mathcal{D}' and \mathcal{C}' are Pr-equivalent.*

Proof (Theorem 1) Statement 1. Assume a HA with decomposed scheduling $\mathcal{D}=(\mathcal{H}, \mathbb{X}, \mathbf{a})$ with $\mathcal{H}=(Loc, Var, Flow, \top, Edge, Init)$, $Loc=\{\ell_1, \dots, \ell_n\}$, $Var=\{v_1, \dots, v_d\}$ and $\mathbb{X}=\{X_1, \dots, X_k\}$. We construct a HA with composed scheduling $\mathcal{C}=(\mathcal{H}', \Psi_c, \Psi_d)$ with $\mathcal{H}'=(Loc, Var', Flow', \top, Edge', Init')$ as follows.

We encode into the state of \mathcal{H}' for each random variable (i) the state in which it was sampled the last time and (ii) the time which has evolved since then. We also encode (iii) the time spent in the current location since last entering.

To account for (i), we introduce fresh variables $D=\{d_i \mid 1 \leq i \leq k\}$ to store the location components and $C=\{c_{i,j} \mid 1 \leq i \leq k, 1 \leq j \leq d\}$ to store the variable values before the last sampling of X_i . Hence, for the i -th random variable $X_i \in \mathbb{X}$, we encode the state of its last (re)sampling by the values of $(d_i, (c_{i,1}, \dots, c_{i,d}))$.

To encode (ii), we introduce k variables $R=\{r_i \mid 1 \leq i \leq k\}$ which capture the time since the last (re)sampling of each random variable $X_i \in \mathbb{X}$ in r_i .

Finally, for (iii) we use t_{jump} to store the time duration since the last jump.

Thus our encoding uses $d' = d + k \cdot (d + 2) + 1$ variables ordered as $Var' = \{v_1, \dots, v_d, d_1, \dots, d_k, c_{1,1}, \dots, c_{k,d}, r_1, \dots, r_k, t_{jump}\}$. For $\nu \in \mathbb{R}^{d'}$, we use $\nu \downarrow_d$ to denote the first d components (ν_1, \dots, ν_d) of $\nu = (\nu_1, \dots, \nu_{d'}) \in \mathbb{R}^{d'}$. Furthermore, for any $\nu \in \mathbb{R}^{d'}$, $a \in Var'$ and $b \in \mathbb{R}$, by $\nu[a \mapsto b]$ we denote ν after changing the entry at the position of variable a (as defined in Var') to b .

The above encoding is implemented by extending $Flow$ to $Flow'$ in each location $\ell \in Loc$ with derivative 0 for each variable in $C \cup D$ and derivative 1 for each variable in $R \cup \{t_{jump}\}$. I.e., for each $\ell \in Loc$ and $\nu \in \mathbb{R}^{d'}$, $Flow'(\nu) = (Flow(\nu \downarrow_d), \mathbf{0}, \mathbf{1})$ with $\mathbf{0}$ is a sequence of $k(d+1)$ zeros and $\mathbf{1}$ of $k+1$ ones.

Further, $Edge' = \{e' \mid e \in Edge\}$ contains for each $e = (\ell_{j_1}, g, r, \ell_{j_2}) \in Edge$ with $\mathbf{a}(e) = i$ the jump $e' = (\ell_{j_1}, g', r', \ell_{j_2})$ which extends e to handle the new variables; formally, $g' = \{\nu \in \mathbb{R}^{d'} \mid \nu \downarrow_d \in g\}$ and for all $\nu \in \mathbb{R}^{d'}$, $r'(\nu) = \nu[v_1 \mapsto r(\nu)[1]] \dots [v_d \mapsto r(\nu)[d]][d_i \mapsto j_2][c_{i,1} \mapsto r(\nu)[1]] \dots [c_{i,d} \mapsto r(\nu)[d]][r_i \mapsto 0][t_{jump} \mapsto 0]$.

For each $\ell \in Loc$, $Init'(\ell)$ consist of all $\nu \in \mathbb{R}^{d'}$ for which $\nu \downarrow_d \in Init(\ell)$, and such that $\nu(d_i) = \ell$, $\nu(c_{i,j}) = \nu(v_j)$ and $\nu(r_i) = \nu(t_{jump}) = 0$ for all $i = 1, \dots, k$ and $j = 1, \dots, d$.

Now we define the kernel Ψ_c . With decomposed scheduling, the duration between two samplings of a random variable (defined by its realisation) might cover consecutive stays in different locations. However, in the composed setting, we are forced to sample a new duration upon entering a new location.

For each state $\sigma = (\ell, \nu) \in \Sigma_{\mathcal{C}}$ let $\sigma_{X_i} = (\ell_{X_i}, \nu_{X_i}) \in \Sigma_{\mathcal{D}}$ denote the state in which X_i was (re)sampled the last time, as encoded in the values of the auxiliary variables: $\ell_{X_i} = \ell_{\nu(d_i)}$ and $\nu_{X_i}(v_j) = \nu(c_{i,j})$ for $j = 1, \dots, d$. For each random variable $X_i \in \mathbb{X}$ with CDF $F_{X_i}^{\sigma_{X_i}}$ and density function $f_{X_i}^{\sigma_{X_i}}$ in state σ_{X_i} , we first define another random variable X'_i whose probability distribution is first conditioned in that samples are at least as large as the time $\nu(r_i)$ passed since the last (re)sampling of X_i , and then shifted by $\nu(r_i)$ to the left: $F_{X'_i}^{\sigma}(x) =$

$$Pr(X_i \leq \nu(r_i) + x \mid X_i > \nu(r_i)) = \frac{\int_{\nu(r_i)}^{\nu(r_i)+x} f_{X_i}^{\sigma_{X_i}}(t) dt}{1 - F_{X_i}^{\sigma_{X_i}}(\nu(r_i))}. \text{ Let } \mathbb{X}' = \{X'_i \mid X_i \in \mathbb{X}\}.$$

For each first state in a (sub)-trace of \mathcal{D} and \mathcal{C} , the probability distribution over the delay must be the same in both automata. We let Ψ_c specify for each state $\sigma \in \Sigma_{\mathcal{C}}$ a probability distribution over the time delay until the next random variable $X_i \in \mathbb{X}$ expires, i.e. $\text{Dist}_{\sigma}^{\Psi_c} = f_M$, where the random variable $M = \min(\mathbb{X}')$ is the minimum over all shifted random variables, as defined in [24].

For the kernel Ψ_d , we observe that for each random variable X_i , in each state $\sigma_{\mathcal{D}}$ exactly one X_i -labeled jump is enabled in \mathcal{D} . Our construction of Edge' maintains this characteristics in \mathcal{C} . To formalize the probability that an enabled jump is taken, we define for each random variable X_i another random variable X_i'' , $\mathbf{X}'' = \{X_1'', \dots, X_k''\}$, which is defined as X_i' but its CDF $F_{X_i''}^{\sigma}(x)$ is shifted with $\nu(r_i) - \nu(t_{\text{jump}})$, instead of $\nu(r_i)$, to model the probabilities of samples beyond the time point of the last jump (i.e. in the definition of $F_{X_i}^{\sigma}(x)$ we replace $\nu(r_i)$ by $\nu(r_i) - \nu(t_{\text{jump}})$). We let the discrete kernel define for each state $\sigma_{\mathcal{C}}$ and each edge $e' \in \text{Edge}'$ with $\text{a}(e') = i$ the probability

$$\Psi_d(\sigma_{\mathcal{C}})(e') = \begin{cases} Pr_{\mathcal{C}}^{\sigma_{\mathcal{C}}}(X_i'' \leq \min(\mathbb{X}'' \setminus \{X_i''\})) & \text{if } e' \text{ is enabled in } \sigma_{\mathcal{C}}, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

Hence, given an arbitrary HA \mathcal{D} with decomposed scheduling, we can construct a HA \mathcal{C} with composed scheduling such that \mathcal{D} and \mathcal{C} have the same trace set and the same initial trace set, and such that each common trace τ is Pr-equivalent in \mathcal{D} and \mathcal{C} . As furthermore Ψ_c is specified such that it mimics the distribution over the duration until the next random variable expires, it is assured that $Pr_{\mathcal{D}}^{\sigma}(X \leq t) = Pr_{\mathcal{C}}^{\sigma}(X \leq t)$.

Statement 2. To show statement 2 of Theorem 1 we consider Figure 3 as a counterexample. The depicted HA with composed scheduling is constructed such that in the initial state a delay distributed according to the uniform distribution $\mathcal{U}(0, 10)$ is sampled, before a jump to location ℓ_1 is taken with probability 0.25 and a jump to location ℓ_2 with probability 0.75.

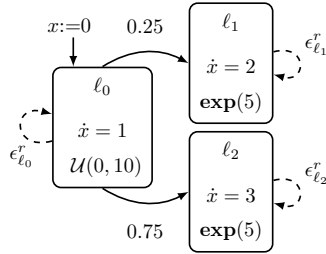


Fig. 3: Counterexample. □

For decomposed scheduling, we associate each edge with an independent random variable X_1 resp. X_2 , where $\min(X_1, X_2)$ should be uniformly distributed over the intersection of the support of X_1 and X_2 in location ℓ_0 to achieve Pr-equivalence. However, the minimum of two continuous random variables with overlapping support can never be uniformly distributed, see [24].

3.5 Extending (De-)composed Scheduling with Invariants

If we allow non-trivial invariants, unrealizable time durations might have a positive probability. To manage such cases, we use the concept of *forced jumps* for both, composed and decomposed scheduling. Forced jumps ensure that no

time step larger than $t_{max}(\sigma)$ is executed. For both, composed and decomposed scheduling, the HA is adapted such that each location has a forced jump which is used to leave a location before its invariant is violated. Furthermore, the semantics for composed and decomposed scheduling (Definitions 4 and 6) are altered such that for each state σ , the time delay is capped by $t'_i = \min(t_i, t_{max}(\sigma))$.

As the probability mass of sampling a delay larger than $t_{max}(\sigma)$ in state σ is shifted to the forced jumps, this has to be considered in the probability measures of Section 3.3 by integrating over $(t_{max}(\sigma), \infty)$ in case of a forced jump.

4 Classification of Existing Approaches

This paper allows for a broad classification covering multiple formalisms. Representatives from literature can be found for both variants, as indicated in Table 1. We discuss one representative formalism for composed and one for decomposed scheduling in more depth. Furthermore, we discuss which formalisms do not fit into our classification and relate our comparison to the work in [17] which focuses on Markovian stochastic hybrid models.

Composed Scheduling. Lygeros and Prandini [15] introduce a general class of *continuous-time stochastic hybrid automata* (CTSHA). This approach has been abstracted to discrete-time e.g., in [2, 20]. CTSHA implement composed scheduling, as the stochastic information over the delay is attached to the location and a stochastic kernel chooses the jump-successor state randomly. Technically, the actual jump times are the stopping time of the inhomogeneous Poisson process with state-dependent rate $\lambda(t) = \lambda(q(t), \mathbf{x}(t))$. This results in delays which are sampled according to an inhomogeneous exponential distribution which can explicitly be expressed by the stochastic kernel Ψ_c used for composed scheduling by: $Pr(X > t \mid (\ell(s), \mathbf{x}(s))) = e^{-\int_s^{s+t} \lambda(u) du}$. The integral can be computed, if for each location, the continuous state evolves with a deterministic rate. Thus, $(\ell(s + t'), \mathbf{x}(s + t'))$ is well defined for any $t' \leq t \in \mathbb{R}_{\geq 0}$ and given $(\ell(s), \mathbf{x}(s))$.

The inhomogeneous Poisson process used in CTSHA can define a phase-type distribution which can approximate any continuous probability distribution. Hence, just as for composed scheduling, the sampled delay can follow any probability distribution in CTSHA. Moreover, CTSHA directly extend our proposed HA with composed scheduling by including stochastic differential equations (SDEs) to describe the continuous state's evolution. Furthermore, the initial state is sampled according to a probability measure and the kernel Ψ_d is extended to a continuous stochastic kernel, enabling random resets of the continuous state.

Table 1: Classification of existing formalisms.

	[2]	[3]	[4, 5]	[6]	[8]	[9]	[15]	[16, 19]	[20]
<i>composed</i>	✓	✓	✓			✓	✓		✓
<i>decomposed</i>				✓	✓	✓		✓	

Decomposed Scheduling. Pilch et al. [16, 19] introduce singular automata with random clocks (SARC) which basically apply decomposed scheduling. They implement this approach by adding *random clocks* which induce random variables characterizing the delay for each jump. Upon expiration, a new random variable is induced which follows the predefined probability distribution with a constant parameter of the corresponding random clock. Additionally, SARC allow for transitions which are not associated with the expiration of a random variable which adds non-determinism to the model. The concept of random clocks can be extended to other sub-classes of hybrid automata, e.g., rectangular automata [6] with random clocks, which again implicitly assign random variables to jumps via random clocks and restrict the syntax to ensure a sound probability measure. In contrast, decomposed scheduling allows to directly attach random variables to jumps and the semantics ensures that the resulting probability measure is sound. This simplifies modeling.

Markovian Stochastic Hybrid Models. The formalism of *Continuous-Time Markov Chains* (CTMCs) which are discrete-state models without variables, has been extended in the past to stochastic hybrid models [4, 5, 8, 12], which correspond to (different kinds of) Markov processes. *Piecewise Deterministic Markov Processes* (PDMPs) [4, 5] implement composed scheduling where the evolution of a continuous state is piece-wise deterministic and can be restricted by invariants. In PDMPs the jump times are ruled by an inhomogenous Poisson process and jumps and their effects are chosen probabilistically by a transition kernel. This stochastic kernel allows to encode guards implicitly.

In contrast, *Switching Diffusion Processes* (SDPs) [8] describe the continuous variables' evolution via stochastic differential equations. They do not allow for invariants or resets of the continuous state and the discrete state evolves according to a *controlled Markov chain* [13], whose transition matrix depends on the continuous state. This allows the user to encode guards into the model. Due to the underlying Markov chain, which can be characterised via a generator matrix describing competing random variables, SDPs can be seen as an approach with decomposed scheduling. *Stochastic hybrid systems* (SHSs) [12] simplify CT-SHA as discussed in [15] by relaxing the inhomogenous Poisson process which determines the random delays.

The formalisms mentioned above coincide under certain assumptions, as discussed in [17]. For example, the authors state that the formalism SDP, which is decomposed in our classification, and the formalism SHS, which we classify as composed, coincide iff invariants and guards of the SHS evaluate to true.

Clearly, restricting to exponentially distributed delays renders the counterexample of Theorem 1 invalid, as the minimum of two exponentially distributed random variables is again exponentially distributed. However, additional restrictions are necessary to ensure that the probability spaces of (de-)composed scheduling coincide in the presence of guards and invariants. Specifying such restrictions (and proving their correctness) is out of scope for this paper.

We refer to [17] for a more detailed comparison on the expressivity of Markovian stochastic hybrid models.

Other Formalisms. Several existing formalisms for stochastic hybrid models do not fit into the proposed classification, as they focus e.g., solely on randomly distributed initial states [18] or on a non-deterministic choice over discrete probability distributions for choosing a successor state [21].

The formalism presented in [3] defines a fully stochastic semantics for timed automata by randomly choosing delays and jumps. It applies a composed semantics, however without resampling jumps. Timelock-freedom is ensured by restricting the support of the probability distributions to executable samples.

The formalism presented in [9] proposes networks of stochastic hybrid automata which fit in both, the composed as well as the decomposed approach. Such stochastic hybrid automata allow to reset continuous variables to realizations of continuous random variables. Thus, at each jump we can either sample a randomly distributed delay for the location like in composed scheduling, or associate the samples as delays with the jumps as in decomposed scheduling.

5 Conclusion

In this paper we formalized two approaches to extend hybrid automata with stochasticity. The first approach applies *composed scheduling*, where two stochastic kernels are used to sample the lengths of time steps and the successor states of jumps. In contrast, the second approach yields *decomposed scheduling* via competing random variables. As the realisations of the random variables specify the delay after which the corresponding jump is taken, a race-condition is induced. The minimum realisation of the random variables then fully characterises the next execution step. We formalized the syntax and semantics for (de-)composed scheduling and the stochastic processes underlying the different resulting models. We defined *trace probability equivalence* and showed that it is possible to construct for every given HA with decomposed scheduling, an equivalent HA with composed scheduling. Via a simple counterexample, we showed that a HA with composed scheduling exists, for which no equivalent HA with decomposed scheduling can be constructed.

To connect the theoretical constructs developed in this paper to existing formalisms, we classified several existing formalisms according to their semantics and pointed to approaches which we cannot capture yet. To include them in our classification, future work will consider more expressive systems, e.g. including stochastic resets and stochastic noise. Furthermore, we plan to investigate the relation of the presented classes to approaches without resampling.

Acknowledgements. We thank the ARCH competition 2023 for fruitful discussions on expressing example models from the ARCH report [1] in the formalism of Lygeros et al. [15].

A Minimum of Two Random Variables

Let A and B be two independent continuous random variables with known CDF $F_A(m)$ and $F_B(m)$, as well as known PDF $f_A(m), f_B(m)$. We then consider the random variable $M = \min(A, B)$. The CDF of the random variable M is given by:

$$\begin{aligned}
 Pr(M \leq m) &= 1 - Pr(M > m) \\
 &= 1 - Pr(A > m, B > m) \\
 &= 1 - (1 - Pr(A \leq m)) \cdot (1 - Pr(B \leq m)) \\
 &= 1 - (1 - F_A(m)) \cdot (1 - F_B(m)) \\
 &= F_A(m) - F_B(m) + F_A(m)F_B(m).
 \end{aligned}$$

Hence, the CDF of M is $F_M(m) = F_B(m) - F_A(m) + F_A(m)F_B(m)$. By taking the derivative of $F_M(m)$ we obtain its PDF $f_M(m)$:

$$\begin{aligned}
 f_M(m) &= \frac{\delta F_M(m)}{\delta m} = f_A(m) + f_B(m) - f_A(m)F_B(m) - f_B(m)F_A(m) \\
 &= f_A(m) + f_B(m) - f_A(m)(1 - Pr(B \geq m)) - f_B(m)(1 - Pr(A \geq m)) \\
 &= f_A(m)Pr(B \geq m) + f_B(m)Pr(A \geq m) \\
 &= f_A(m) \int_m^\infty f_B(m)dt + f_B(m) \int_m^\infty f_A(m)dt.
 \end{aligned}$$

Hence, the probability density function of M is $f_M(m) = f_A(m) \int_m^\infty f_B(m)dt + f_B(m) \int_m^\infty f_A(m)dt$.

In the following, we require that $supp(f_A) \cap supp(f_B) \neq \emptyset$. In the following we show, that $M = \min(A, B)$ cannot be uniformly distributed.

Rearranging terms we obtain:

$$\begin{aligned}
 f_M(m) &= f_A(m) \int_m^\infty f_B(m)dt + f_B(m) \int_m^\infty f_A(m)dt \\
 &= f_A(m)(1 - F_B(m)) + f_B(m)(1 - F_A(m)) \\
 &= f_A(m) - f_A(m)F_B(m) + f_B(m) - f_B(m)F_A(m).
 \end{aligned}$$

Since A, B are continuous random variables, $f_i(m) \geq 0$, $F_i(m)$ continuous and monotone increasing with $\lim_{m \rightarrow \infty} F_i(m) = 1$ and $\lim_{m \rightarrow -\infty} F_i(m) = 0$, for $i \in \{A, B\}$. Hence, $m_1, m_2 \in supp(f_A) \cap supp(f_B)$ exist, for which $0 \leq F_A(m_1) < F_A(m_2) \leq 1$ and $0 \leq F_B(m_1) < F_B(m_2) \leq 1$ and hence

$$\begin{aligned}
 F_A(m_1) < F_A(m_2) &\Rightarrow f_B(m_1)F_A(m_1) < f_B(m_2)F_A(m_2) \\
 &\Rightarrow f_B(m_1) - f_B(m_1)F_A(m_1) > f_B(m_2) - f_B(m_2)F_A(m_2), \text{ and} \\
 F_B(m_1) < F_B(m_2) &\Rightarrow f_A(m_1)F_B(m_1) < f_A(m_2)F_B(m_2) \\
 &\Rightarrow f_A(m_1) - f_A(m_1)F_B(m_1) > f_A(m_2) - f_A(m_2)F_B(m_2).
 \end{aligned}$$

Thus, $f_M(m_1) \neq 0$ and $f_M(m_2) \neq 0$, with $f_M(m_1) \neq f_M(m_2)$ and therefore f_M cannot be a uniform distribution.

References

1. Abate, A., Blom, H., Delicaris, J., Haesaert, S., Hartmanns, A., van Huijgevoort, B., Lavaei, A., Ma, H., Niehage, M., Remke, A., Schön, O., Schupp, S., Soudjani, S., Willemsen, L.: Arch-comp22 category report: Stochastic models. In: Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22). EPiC Series in Computing, vol. 90, pp. 113–141. EasyChair (2022). <https://doi.org/10.29007/lsvc>
2. Abate, A., Prandini, M., Lygeros, J., Sastry, S.: Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica* **44**(11), 2724–2734 (2008). <https://doi.org/10.1016/j.automatica.2008.03.027>
3. Bertrand, N., Bouyer, P., Brihaye, T., Menet, Q., Baier, C., Größer, M., Jurdzinski, M.: Stochastic timed automata. *Logical Methods in Computer Science* **10**(4) (2014). [https://doi.org/10.2168/LMCS-10\(4:6\)2014](https://doi.org/10.2168/LMCS-10(4:6)2014)
4. Bujorianu, M.L., Lygeros, J.: Reachability questions in piecewise deterministic Markov processes. In: Proc. of Hybrid Systems: Computation and Control (HSCC’03). pp. 126–140. Springer (2003). https://doi.org/10.1007/3-540-36580-X_12
5. Davis, M.H.: *Markov Models & Optimization* (1st Edition). Routledge (1993). <https://doi.org/10.1201/9780203748039>
6. Delicaris, J., Schupp, S., Ábrahám, E., Remke, A.: Maximizing reachability probabilities in rectangular automata with random clocks. In: *Theoretical Aspects of Software Engineering - 17th International Symposium, TASE 2023. Lecture Notes in Computer Science*, vol. 13931, pp. 164–182. Springer (2023). https://doi.org/10.1007/978-3-031-35257-7_10
7. Fränzle, M., Hahn, E.M., Hermanns, H., Wolovick, N., Zhang, L.: Measurability and safety verification for stochastic hybrid systems. In: Proc. of the 14th ACM Int. Conf. on Hybrid Systems: Computation and Control (HSCC’11). pp. 43–52. ACM (2011)
8. Ghosh, M.K., Arapostathis, A., Marcus, S.I.: Ergodic control of switching diffusions. *SIAM Journal on Control and Optimization* **35**(6), 1952–1988 (1997). <https://doi.org/10.1137/S0363012996299302>
9. Hahn, E.M., Hartmanns, A., Hermanns, H., Katoen, J.: A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design* **43**(2), 191–232 (2013). <https://doi.org/10.1007/s10703-012-0167-z>
10. Henzinger, T.A.: The theory of hybrid automata. In: *Verification of Digital and Hybrid Systems*, pp. 265–292. Springer (2000). <https://doi.org/10/dpjwvs>
11. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What’s decidable about hybrid automata? *Journal of Computer and System Science* **57**(1), 94–124 (1998). <https://doi.org/10.1006/jcss.1998.1581>
12. Hu, J., Lygeros, J., Sastry, S.: Towards a theory of stochastic hybrid systems. In: Proc. of the 3rd Int. Workshop on Hybrid Systems: Computation and Control (HSCC’00). pp. 160–173. LNCS, Springer (2000). https://doi.org/10.1007/3-540-46430-1_16
13. Kesten, H., Spitzer, F.: Controlled Markov chains. *The Annals of Probability* **3**(1), 32–40 (1975). <https://doi.org/10.1214/aop/1176996445>
14. Klenke, A.: *Probability Theory: A Comprehensive Course*. Springer, London (2014). https://doi.org/10.1007/978-1-4471-5361-0_{_}1
15. Lygeros, J., Prandini, M.: Stochastic hybrid systems: A powerful framework for complex, large scale applications. *European Journal of Control* **16**(6), 583–594 (2010). <https://doi.org/10.3166/ejc.16.583-594>

16. Pilch, C., Schupp, S., Remke, A.: Optimizing reachability probabilities for a restricted class of stochastic hybrid automata via flowpipe-construction. In: Proc. of the 18th Int. Conf. on Quantitative Evaluation of Systems (QEST'21). LNCS, vol. 12846, pp. 435–456. Springer (2021). https://doi.org/10.1007/978-3-030-85172-9_{2}{3}
17. Pola, G., Bujorianu, M., Lygeros, J., Benedetto, M.D.D.: Stochastic hybrid models: An overview. In: IFAC Conf. on Analysis and Design of Hybrid Systems (ADHS'03). IFAC Proceedings Volumes, vol. 36, pp. 45–50. Elsevier (2003). [https://doi.org/10.1016/S1474-6670\(17\)36405-4](https://doi.org/10.1016/S1474-6670(17)36405-4)
18. Shmarov, F., Zuliani, P.: ProbReach: Verified probabilistic δ -reachability for stochastic hybrid systems. In: Proc. of the 18th ACM Int. Conf. on Hybrid Systems: Computation and Control, (HSCC'15). p. 134–139. HSCC '15, ACM (2015). <https://doi.org/10.1145/2728606.2728625>
19. da Silva, C., Schupp, S., Remke, A.: Optimizing reachability probabilities for a restricted class of stochastic hybrid automata via flowpipe-construction. Transactions on Modeling and Computer Simulation (2023). <https://doi.org/10.1145/3607197>, just accepted
20. Soudjani, S.E.Z., Abate, A.: Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. SIAM Journal on Applied Dynamical Systems **12**(2), 921–956 (2013). <https://doi.org/10.1137/120871456>
21. Sproston, J.: Decidable model checking of probabilistic hybrid automata. In: Proc. of the 6th Int. Symp. on Formal Techniques in Real-Time and Fault-Tolerant Systems (FTRTFT'00). LNCS, vol. 1926, pp. 31–45. Springer (2000). https://doi.org/10.1007/3-540-45352-0_5
22. Sproston, J.: Verification and control of probabilistic rectangular hybrid automata. In: Proc. of the 13th Int. Conf. on Formal Modeling and Analysis of Timed Systems (FORMATS'15). LNCS, vol. 9268, pp. 1–9. Springer (2015). https://doi.org/10.1007/978-3-319-22975-1_1
23. Teige, T., Fränzle, M.: Constraint-based analysis of probabilistic hybrid systems. In: 3rd IFAC Conference on Analysis and Design of Hybrid Systems, (ADHS'09). IFAC Proceedings Volumes, vol. 42, pp. 162–167. Elsevier (2009). <https://doi.org/10.3182/20090916-3-ES-3003.00029>
24. Willemsen, L., Remke, A., Ábrahám, E.: Comparing two approaches to include stochasticity in hybrid automata (2023). <https://doi.org/10.48550/arXiv.2307.08052>