EPiC Series in Computing

Volume 96, 2023, Pages 126–150

Proceedings of 10th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH23)

ARCH-COMP23 Category Report: Stochastic Models

Alessandro Abate[1], Henk Blom[2], Nathalie Cauchi[1], Joanna Delicaris[3], Sofie Haesaert[4], Birgit van Huijgevoort[4], Abolfazl Lavaei[5], Anne Remke[3], Oliver Schön[5], Stefan Schupp[6], Fedor Shmarov[7], Sadegh Soudjani[5], Lisa Willemsen[8], and Paolo Zuliani[9]

[1] University of Oxford, Oxford, UK
[2] Delft University of Technology, Delft, The Netherlands
[3] University of Münster, Münster, Germany
[4] TU Eindhoven, Eindhoven, Netherlands
[5] Newcastle University, Newcastle upon Tyne, UK
[6] TU Wien, Vienna, Austria
[7] University of Manchester, Manchester, UK
[8] University of Twente, Enschede, The Netherlands
[9] Università di Roma "La Sapienza", Rome, Italy

Abstract

This report is concerned with a friendly competition for formal verification and policy synthesis of stochastic models. The main goal of the report is to introduce new benchmarks and their properties within this category and recommend next steps toward next year's edition of the competition. Given that the tools for stochastic models are at their early stages of development compared to those of non-probabilistic models, the main focus is to report on an initiative to collect a set of minimal benchmarks that all such tools can run, thus facilitating the comparison between the efficiency of the implemented techniques. This friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in Summer 2023.

1 Introduction

The subgroup "Stochastic Models" of the annual friendly ARCH-Competition focuses on recent developments of tools that can analyze systems that exhibit uncertain, stochastic behavior. This includes a diverse set of systems, expressing uncertainty in its various ways, e.g., continuously applied stochasticity or discrete mode changes, which happen with a certain probability.

Disclaimer The presented report of the ARCH friendly competition for stochastic modeling group aims at providing a unified point of reference on the current state of the art in the area of stochastic models together with the currently available tools and framework for performing formal verification and optimal policy synthesis to such models. We further provide a set of benchmarks, which we aim to push forward the development of current and future tools. To establish further trustworthiness of the results, the code describing the benchmarks together with the code used to compute the results is publicly available at https://gitlab.com/goranf/ARCH-COMP.

This friendly competition is organized by Alessandro Abate (alessandro.abate@cs.ox.ac.uk), Abolfazl Lavaei (abolfazl.lavaei@newcastle.ac.uk), Stefan Schupp (stefan.schupp@tuwien.ac.at), and Sadegh Soudjani (sadegh.soudjani@newcastle.ac.uk).

This report presents the results of the ARCH Friendly Competition 2023 in the group *stochastic models*. We refer the interested reader to the survey paper [14] and references therein for the details of most of the underlying techniques used in the development of the tools of this category. The following tools and frameworks have participated in this category so far: (in alphabetical order): AMYTISS, FAUST^2, FIGARO workbench, hpnmg, HYPEG, Mascot-SDS, the Modest Toolset, ProbReach, PyCATSHOO, RealySt, SDCPN&IPS, SReachTools, StocHy, and SySCoRe.

Tools participated in this year are (in alphabetical order): HYPEG, ProbReach, RealySt, and SySCoRe. The benchmark collection has been enriched by an extended version of a *Package Delivery* benchmark that differs from the original version in that it shows more complex noise distributions which may pose additional challenges for the tools. This benchmark can be used to check specifications that are expressed by various classes of finite state automata. We have also improved the collection of *minimal examples* first presented last year. The goal of this collection is to provide a set of simple challenges such that different tools can be employed with the least modifications of the underlying model. The initiative for developing this benchmark will allow us to compare the tools that previously were only applicable to separate set of benchmarks.

Similar to last years, all participants were encouraged to provide a repeatability package (e.g., a Docker container) for centralized evaluation on the servers of the ARCH-group. Apart from providing repeatable results, this allows for sharing of the tools themselves to both the ARCH and the wider research community.

This report has the following structure. Section 2 provides a short overview of the participating tools and frameworks. Sections 3, 4 present already established benchmarks and a set of new benchmark descriptions, which include a discussion of the individual models syntax and semantics. In Section 5 we present the results of the friendly competition with the participating tools or algorithmic frameworks that are used to solve instances of the collection of benchmarks. We identify key challenges and discuss future plans in Section 6.

2 Participating Tools & Frameworks

HYPEG The Java-based library HYPEG [21] implements time-bounded discrete-event simulation for hybrid Petri nets with general transitions (HPnGs) [5], which combine discrete and continuous components with a possibly large number of random variables, whose stochastic behavior follows arbitrary probability distributions. HYPEG uses well-known statistical model checking techniques to verify complex properties, including time-bounded reachability [22]. These techniques comprise several hypothesis tests as well as different approaches for the computation of confidence intervals. Continuous behavior that can be expressed by systems of ordinary differential equations can be simulated using an approximative approach [20, 18], whereas piecewise-linear

continuous behavior is simulated without approximation. HYPEG resolves discrete nondeterminism either probabilistically or using reinforcement learning to maximize or minimize the probability of a property [17, 19], also in combination with a contract-based approach [2]. The tool is available at https://zivgitlab.uni-muenster.de/ag-sks/tools/HYPEG.

ProbReach ProbReach [26] implements algorithms for computing probabilistic bounded reachability in so-called *parametric* stochastic hybrid systems. These models may feature nonlinear continuous dynamics (i.e., defined by nonlinear ODEs) and random (continuous and discrete) and/or nondeterministic parameters. ProbReach can compute numerically sound *enclosures* that are formally guaranteed to contain the exact value of the bounded reachability probability [27] (for models with only random parameters the enclosure's size can be made arbitrarily small). Furthermore, ProbReach implements Monte Carlo simulation algorithms [25] for computing confidence intervals for the bounded reachability probability with numerically sound sampling (i.e., the model simulations are numerically rigorous). ProbReach is written in C++ and is available at https://github.com/dreal/probreach.

RealySt This tool optimizes reachability probabilities for the class of rectangular automata with random clocks (c.f. [3]), which exhibit discrete and continuous nondeterminism as well as stochasticity. Using forward reachability anaylsis and a backward refinement approach, probabilities can be maximized [3]. It is implemented in C++ and relies on the library HyPro [24] for the state-set representation via convex polytopes as well as efficient geometric operations. Monte Carlo integration algorithms for multi-dimensional integration are provided by the GNU Scientific Library (GSL) [4] and improved for high-dimensional polytopes [28]. RealySt builds on the tool hpnmg [9], a model checker for Hybrid Petri nets with an arbitrary but finite number of general transition firings for specifications formulated in STL [10, 12, 13, 11] and on the flowpipe-based approach to optimize reachability probabilities in singular automata with random clocks [23]. RealySt is currently being developed within the DFG project 471367371 as a cooperation between the RWTH Aachen and the University of Münster and is available at https://zivgitlab.uni-muenster.de/ag-sks/tools/realyst.

SySCoRe This tool stands for Synthesis via Stochastic Coupling Relations for stochastic continuous state systems and is a MATLAB toolbox for temporal logic control synthesis of discrete-time continuous-state stochastic dynamical systems [8]. The fully automated implementation starts from a system description and a temporal logic specification and computes a robust controller alongside robust quantified bounds on the probability of satisfying the given property based on space discretization. The tool is based on establishing simulation relations between stochastic processes based on coupling the underlying stochastic distributions. The developed algorithms compute two precision parameters (ϵ, δ), which bound the deviations in both the output trajectories (ϵ) and the transition probabilities (δ) of the models. The development of SySCoRe is mainly based on the papers [6, 7]. The main advantage of SySCoRe compared to alternative tools is the fact that the computed error does not grow linearly in time, which makes the tool applicable to infinite horizon properties and unbounded disturbances. The current version of SySCoRe supports nonlinear dynamics, complex co-safe temporal logic specifications over infinite horizons, model-order reduction, arbitrary (possibly unbounded) additive disturbance, and fast tensor computations.

It is worth noticing that further tools participated in previous editions (see e.g., [1]). Table 1 shows all tools that participated and the years in which they solved a benchmark the first time.

3 Established Benchmarks

In the previous editions of ARCH, we have presented numerous benchmarks for the sake of friendly competitions. These benchmarks include automated anesthesia (AS), building automation (BA), heated tank (HT), water sewage (WS), stochastic Van der Pol (VP), integrator chain (IC), autonomous vehicle (AV), patrol robot (PR), geometric Brownian motion (GB), minimal examples (ME), and package delivery (PD). In Table 1, we indicate the year a tool was first applied to a given benchmark. We also present an overview of benchmark properties in Table 2.

Table 1: Tool-benchmark matrix: We indicate the year a tool was first applied to a given benchmark. Shortkeys: automated anesthesia (AS), building automation (BA), heated tank (HT), water sewage (WS), stochastic Van der Pol (VP), integrator chain (IC), autonomous vehicle (AV), patrol robot (PR), geometric Brownian motion (GB), minimal examples (ME), package delivery (PD), extended package delivery (PDx).

Tool	Benchmarks											
	AS	BA	HT	WS	VP	IC	AV	PR	GB	ME	PD	PDx
FAUST^2	2018	2018				2020						
StocHy	2019	2019				2020						
SReachTools	2018	2018				2020						
AMYTISS	2020	2020			2020	2020	2020		2021			
hpnmg				2020								
HYPEG			2019	2020						2022		
Mascot-SDS					2020			2021				
modes			2018	2020						2022		
ProbReach				2020								
prohver			2020	2020						2022		
RealySt										2022		
SDCPN&IPS			2019						2021			
SySCoRe		2021			2022						2022	2023
FIGARO workbench			2021									
PyCATSHOO			2021									

4 Extended Benchmarks

In this section, we present novel benchmarks or variants of old benchmarks that have been proposed during previous editions of this competition (see, e.g., [1] for further benchmarks), allowing for new outcomes.

4.1 Package Delivery (extended)

We extend the original version of the package delivery case study introduced in [1], which is limited to Gaussian noise distributions, to arbitrary continuous distributions that can be approximated with *Gaussian mixture models* (GMM) [16]. With this, the case study aims at showing if the tools can synthesize controllers for both complex specifications, i.e., for non-acyclic DFAs, and complex noise distributions.

Consider a simple time-discrete system with a continuous state x, control input u, and noise

Table 2: Overview of benchmark properties. Shortkeys: Time horizon: Finite (F) or Infinite (I); Type of control: Switching (S), Drift (Dr), or Multiple (M); Time line: Discrete (D) or Continuous (C); State space: Continuous (C) or Hybrid (H); Drift in ODE/SDE: Linear (L), Piecewise Linear (pL), or Nonlinear (NL); Noise: State-dependent (SD) or fixed (FX), Gaussian (G) or Gaussian mixture model (GM), and Brownian motion (BM) or independently and identically distributed (iid), Rate/Size spont. jumps: State-dependent (SD) or fixed (FX)

Aspect	Benchmarks											
	AS	BA	HT	WS	VP	IC	AV	PR	GB	ME	PD	PDx
Liveness/deadlock					✓			✓				
Prob. reachability	✓	✓	✓	✓		✓	✓		✓	✓	✓	✓
Control synthesis	✓	✓				✓	✓	✓			✓	✓
Min-max		✓					✓					
Time horizon	F	F	F	F	I	F	F	I	F	F	I	I
Type of control	S	M			Dr	Dr	M	M				
Time line	D	D	C	C	D	D	D	D	C	C	D	D
State space	C	H	H	H	C	C	C	H	C	H	C	C
Drift in ODE/SDE	pL	NL	NL	pL	NL	L	NL	NL	L	pL	L	L
Noise in SDE	FX	FX			FX	FX	FX	FX	SD	SD	FX	FX
Noise type in SDE											G	GM
Noise: BM or i.i.d.	iid	iid			iid	iid	iid	iid	BM	iid	iid	iid
Guards		✓	✓	✓			✓			✓		
Rate spont. jumps	FX		SD	FX		FX				SD		
Size spont. jumps	FX		FX	FX		FX				FX		
Environment		✓		✓			✓	✓				
Subsystems		✓	✓	✓								
Concurrency			✓	✓						✓		
Synchronization			✓	✓								
Shared variables		✓		✓								
# discr. states		5	576	35				2		3-5	1	1
# continuous vars.	3	7	2	11	2	50	7	4	1	1-2	2	2
# model params.	24	19	15	36	3	8	11	2	5	7	6	12

w. Assume that the system's dynamics are captured by the following equations:

$$\begin{aligned} \underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}(t)} &= A\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x(t)} + B\underbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}_{u(t)} + \underbrace{\begin{bmatrix}w_1\\ w_2\end{bmatrix}}_{w(t)}, \\ y(t) &= x(t), \end{aligned} \tag{1}$$

with states $x \in \mathbb{X} = [-6, 6] \times [-6, 6]$ and dynamics matrices given by

$$A = \begin{bmatrix}0.9 & 0\\ 0 & 0.8\end{bmatrix}, \quad B = \begin{bmatrix}1.7 & 0\\ 0 & 1.7\end{bmatrix}.$$

The noise w follows a multimodal distribution given by a *homoscedastic* GMM, i.e, $w \sim \sum_{k=1}^{K} \pi_k \mathcal{N}(\cdot|\mu_k, \Sigma)$ with mixing weights $\pi := (\pi_1, \ldots, \pi_K)$, mean values $\mu := (\mu_1, \ldots, \mu_K)$, and a common covariance matrix Σ. The parameters are chosen to be $\pi := (0.8, 0.2)$, $\mu := ([0; 0.8], [-0.8; -0.8])$, and $\Sigma := [\sqrt{0.2}, 0; 0, \sqrt{0.2}]$. We consider a package delivery scenario, for which we define three regions p_1, p_2, and p_3 as follows: $p_1 := [3, 6] \times [-2.5, 1]$, $p_2 := [-1, 1] \times [-4, 3]$

and $p_3 := [-6, -3]^2$. The agent described by the dynamics in (1) can collect a package at p_1 and must deliver it to the target region p_3 (cf. Fig. 1). If the agent visits the avoid region p_2 whilst carrying a package, he loses the package and has to restart by picking up a new package at p_1. This corresponds to the *scLTL* specification $\Diamond(p_1 \wedge (\neg p_2 \mathsf{U} p_3))$, which is captured by the DFA given in Fig. 2. Note that we require the agent to remain in the bounded state space $\mathbb{X}$.

Problem 1. *Compute a policy for the dynamical system in* (1) *that maximizes the probability of satisfying the* scLTL *specification* $\Diamond(p_1 \wedge (\neg p_2 \mathsf{U} p_3))$.

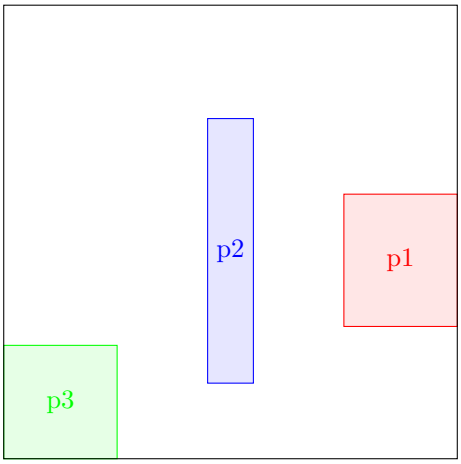

Figure 1: Regions defined over the output space

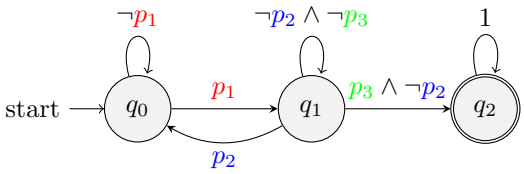

Figure 2: DFA

4.2 Minimal Examples

The idea of this benchmark is to create minimal examples of stochastic hybrid automata, which fit several formalisms. This allows us to compare different model characteristics and see how different tools are able to tackle these. Hence, the following cases include instances of discrete nondeterminism and race conditions between random variables, stochastic noise and time locks. In the following, we present four cases (A, B, C, D) and two variants per case (Variant 1 and Variant 2) as our minimal examples to fit numerous formalisms. Each case/variant is specified in two formalisms: i) the stochastic hybrid automata (SHA) formalism of [15], and ii) the rectangular automata with random clocks (RAR) formalism of [3].

4.2.1 Problem description

To compare how different formalisms and tools realize the four cases, we compute the probability of time-bounded reachability from a unique initial state, i.e. the probability that the continuous variable x reaches a value ≤ -1 before a time bound of 10.

Cases B and C implement an additional time delay of two time units. Here, we compute the reachability probability for an extended time bound of 12, i.e. the probability that the continuous variable x reaches a value ≤ -1 before a time bound of 12.

4.2.2 Case A

The general idea of case A is to model a continuous variable that initially increases and a race condition between two random variables, where the evolution of the continuous variable is stopped, if the first random variable wins the race; and the continuous variable is decreased, if the second random variable wins. The goal is to find the probability that the continuous variable reaches a specific value < 0.

We consider two sets of random variables for all cases: 1) both random variables are distributed exponentially with rates λ_1 and λ_2, and 2) one variable is distributed exponentially with rate λ_1 and the other follows a folded normal distribution with parameters μ and σ.

Variant 1 for case A in the formalism of Lygeros and Prandini Case A can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_A = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, \mathcal{R})$ is a SHA, with

- $\mathcal{Q} = \{\ell_0, \ell_1, \ell_2\}$ discrete states
- $\mathcal{X} = \mathbb{R}$ continuous states
- $Dom(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom(\ell_i) = \mathbb{R}$ for $i \in \{0, 1, 2\}$
- $f : S \to \mathbb{R}$ is a vector field:
  - $f(\ell_0, x) = 2$
  - $f(\ell_1, x) = 0$
  - $f(\ell_2, x) = -3$
- $g : S \to \mathbb{R}^{1\times 1}$ is a diffusion coefficient: $g(\ell_i, x) = 0$ for $i \in \{0, 1, 2\}$
- $Init : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init(\ell_0, 0) = 1$
  - $Init(a) = 0$ for any other $a \in S$
- $\lambda : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda(\ell_0, x) = \lambda_1 + \lambda_2$
  - $\lambda(\ell_i, x) = 0$ for $i \in \{1, 2\}$
- $\mathcal{R} : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:
  - $\mathcal{R}\Big(\big(\ell_0, x\big), \big\{\big(\ell_i, x\big)\big\}\Big) = \frac{\lambda_i}{\lambda_1+\lambda_2}$ for $i \in \{1, 2\}$
  - $\mathcal{R}\Big(\big(\ell_i, x\big), \big\{\big(\ell_i, x\big)\big\}\Big) = 1$ for $i \in \{1, 2\}$
  - $\mathcal{R}\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $S, \bar{S}$ as defined in [15].

Variant 2 for case A in the formalism of Lygeros and Prandini Case A can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_A = (\mathcal{Q}, \mathcal{X}', Dom', f', g', Init', \lambda', \mathcal{R}')$ is a SHA, with

- $\mathcal{X}' = \mathbb{R}^2$ continuous states
- $Dom'(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom'(\ell_i) = \mathbb{R}^2$ for $i \in \{0, 1, 2\}$
- $f' : S \to \mathbb{R}^2$ is a vector field:
  - $f'(\ell_0, (t, x)) = (1, 2)$
  - $f'(\ell_1, (t, x)) = (1, 0)$
  - $f'(\ell_2, (t, x)) = (1, -3)$
- $g' : S \to \mathbb{R}^{2\times 1}$ is a diffusion coefficient: $g'(\ell_i, (t, x)) = (0, 0)$ for $i \in \{0, 1, 2\}$
- $Init' : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init'(\ell_0, (0, 0)) = 1$
  - $Init'(a) = 0$ for any other $a \in S$
- $\lambda' : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda'(\ell_0, (t, x)) = \lambda_1 + \lambda_2(t)$
  - $\lambda'(\ell_i, (t, x)) = 0$ for $i \in \{1, 2\}$
- $\mathcal{R}' : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:
  - $\mathcal{R}'\Big(\big(\ell_0, (t, x)\big), \big\{\big(\ell_2, (t, x)\big)\big\}\Big) = \frac{\lambda_i(t)}{\lambda_1(t)+\lambda_2(t)}$ for $i \in \{1, 2\}$ and $\lambda_1(t) = \lambda_1$
  - $\mathcal{R}'\Big(\big(\ell_i, (t, x)\big), \big\{\big(\ell_i, (t, x)\big)\big\}\Big) = 1$ for $i \in \{1, 2\}$
  - $\mathcal{R}'\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2, \lambda_2(t) = \frac{p_{\mathcal{N}_f}(t)}{\int_t^\infty p_{\mathcal{N}_f}(x)dx}$. Here, $p_{\mathcal{N}_f}$ is the PDF of a folded normal distribution $\mathcal{N}_f(\mu, \sigma^2)$. Furthermore, $S, \bar{S}$ are defined as in [15] and $\mathcal{Q}$ is defined in variant 1.

Variant 1 for case A in the formalism of Delicaris et al. Case A can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (Loc, Var_C, Var_R, Distr, Inv, Init, Flow_C, Flow_R, Edge_C, Edge_R)$ is a RAR, with

- $Loc = \{\ell_0, \ell_1, \ell_2\}$ is a finite set of locations
- $Var_C = \{x\}$ is a finite ordered set of variables
- $Var_R = \{r_1, r_2\}$ is a finite ordered set of random clocks
- $Distr : Var_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $Distr(r_1) = Exp(\lambda_1)$

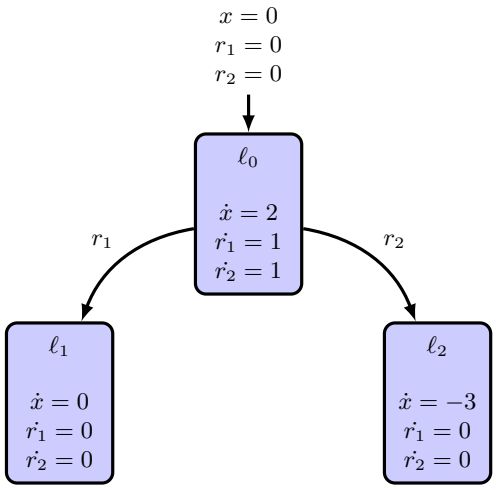

Figure 3: Case A modelled as a rectangular automaton with random clocks (RAR).

  - $Distr(r_2) = Exp(\lambda_2)$

- $Inv : Loc \to \mathbb{I}^1$ is a function assigning an invariant to each location:
  - $Inv(\ell_i) = \mathbb{R}$ for $i \in \{0, 1, 2\}$
- $Init : Loc \to \mathbb{I}^1$ is a function assigning initial states to each location:
  - $Init(\ell_0) = [0, 0]$
  - $Init(\ell_i) = \emptyset$ for $i \in \{1, 2\}$
- $Flow_C : Loc \to \mathbb{I}^1$ is a function assigning a flow to each location:
  - $Flow_C(\ell_0) = [2, 2]$
  - $Flow_C(\ell_1) = [0, 0]$
  - $Flow_C(\ell_2) = [-3, -3]$
- $Flow_R : Loc \to \mathbb{I}^2$ is a function assigning a flow for the random clocks to each location:
  - $Flow_R(\ell_0) = ([1, 1], [1, 1])$
  - $Flow_R(\ell_i) = ([0, 0], [0, 0])$ for $i \in \{1, 2\}$
- $Edge_C = \emptyset \subseteq Loc \times \mathbb{I}^1 \times \mathbb{I}^1 \times 2^{Var_C} \times Loc$ is a finite set of *non-stochastic jumps*
- $Edge_R = \{e_1, e_2\} \subseteq Loc \times Var_R \times Loc$ is a finite set of *stochastic jumps*:
  - $e_1 = (\ell_0, r_1, \ell_1)$
  - $e_2 = (\ell_0, r_2, \ell_2)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 3 for a graphical representation of this model as a RAR.

Variant 2 for case A in the formalism of Delicaris et al. Case A can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (Loc, Var_C, Var_R, Distr', Inv, Init, Flow_C, Flow_R, Edge_C, Edge_R)$ is a RAR, with

- $Distr' : Var_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $Distr'(r_1) = Exp(\lambda_1)$
  - $Distr'(r_2) = \mathcal{N}_f(\mu, \sigma^2)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2$, and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 3 for a graphical representation of this model as a RAR.

4.2.3 Case B

The general idea of case B is to extend both versions of case A with a nondeterministic choice. If a specific random variable wins the stochastic race, a nondeterministic choice occurs after a certain time delay. Note that different formalisms handle (discrete) nondeterminism differently, i.e., Lygeros and Prandini resolve all nondeterminism probabilistically. Hence, using the formal notation of [15] already deviates from the general idea in terms of nondeterministic choices.

Variant 1 for case B in the formalism of Lygeros and Prandini Case B can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_B = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, \mathcal{R})$ is a SHA, with

- $\mathcal{Q} = \{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4\}$ discrete states
- $\mathcal{X} = \mathbb{R}^2$ continuous states
- $Dom(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom(\ell_i) = \mathbb{R}^2$ for $i \in \{0, 1, 3, 4\}$
  - $Dom(\ell_2) = \mathbb{R} \times (-\infty, 2)$
- $f : S \to \mathbb{R}^2$ is a vector field:
  - $f(\ell_0, (x, y)) = (2, 0)$
  - $f(\ell_1, (x, y)) = (0, 0)$
  - $f(\ell_2, (x, y)) = (0, 1)$
  - $f(\ell_3, (x, y)) = (0, 0)$
  - $f(\ell_4, (x, y)) = (-3, 0)$
- $g : S \to \mathbb{R}^{2\times 1}$ is a diffusion coefficient: $g(\ell_i, (x, y)) = (0, 0)$ for $i \in \{0, 1, 2, 3, 4\}$
- $Init : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init(\ell_0, (0, 0)) = 1$
  - $Init(a) = 0$ for any other $a \in S$
- $\lambda : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda(\ell_0, (x, y)) = \lambda_1 + \lambda_2$

  - $\lambda(\ell_i,(x,y)) = 0$ for $i \in \{1,2,3,4\}$

- $\mathcal{R} : \bar{S} \times \mathcal{B}(S) \to [0,1]$ is a transition measure:
  - $\mathcal{R}\Big(\big(\ell_0,(x,y)\big), \big\{\big(\ell_i,(x,y)\big)\big\}\Big) = \frac{\lambda_i}{\lambda_1+\lambda_2}$ for $i \in \{1,2\}$
  - $\mathcal{R}\Big(\big(\ell_2,(x,y)\big), \big\{\big(\ell_i,(x,y)\big)\big\}\Big) = \begin{cases} \frac{1}{2}, & \text{if } y = 2 \\ 0, & \text{else} \end{cases}$ for $i \in \{3,4\}$
  - $\mathcal{R}\Big(\big(\ell_i,(x,y)\big), \big\{\big(\ell_i,(x,y)\big)\big\}\Big) = 1$ for $i \in \{1,3,4\}$
  - $\mathcal{R}\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $S, \bar{S}$ as defined in [15].

Variant 2 for case B in the formalism of Lygeros and Prandini Case B can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_B = (\mathcal{Q}, \mathcal{X}', Dom', f', g', Init', \lambda', \mathcal{R}')$ is a SHA, with

- $\mathcal{X}' = \mathbb{R}^3$ continuous states
- $Dom'(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom'(\ell_i) = \mathbb{R}^3$ for $i \in \{0,1,3,4\}$
  - $Dom'(\ell_2) = \mathbb{R}^2 \times (-\infty, 2)$
- $f' : S \to \mathbb{R}^3$ is a vector field:
  - $f'(\ell_0,(t,x,y)) = (1,2,0)$
  - $f'(\ell_1,(t,x,y)) = (1,0,0)$
  - $f'(\ell_2,(t,x,y)) = (1,0,1)$
  - $f'(\ell_3,(t,x,y)) = (1,0,0)$
  - $f'(\ell_4,(t,x,y)) = (1,-3,0)$
- $g' : S \to \mathbb{R}^{3\times 1}$ is a diffusion coefficient: $g'(\ell_i,(t,x,y)) = (0,0,0)$ for $i \in \{0,1,2,3,4\}$
- $Init' : \mathcal{B}(S) \to [0,1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init'(\ell_0,(0,0,0)) = 1$
  - $Init'(a) = 0$ for any other $a \in S$
- $\lambda' : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda'(\ell_0,(t,x)) = \lambda_1 + \lambda_2(t)$, with $\lambda_2(t) = \frac{p_{\mathcal{N}_f}(t)}{\int_t^\infty p_{\mathcal{N}_f}(x)dx}$
  - $\lambda'(\ell_i,(t,x,y)) = 0$ for $i \in \{1,2,3,4\}$
- $\mathcal{R}' : \bar{S} \times \mathcal{B}(S) \to [0,1]$ is a transition measure:
  - $\mathcal{R}'\Big(\big(\ell_0,(t,x,y)\big), \big\{\big(\ell_i,(t,x,y)\big)\big\}\Big) = \frac{\lambda_i(t)}{\lambda_1(t)+\lambda_2(t)}$ for $i \in \{1,2\}$ and $\lambda_1(t) = \lambda_1$

- $\mathcal{R}'\Big(\big(\ell_2,(t,x,y)\big),\big\{\big(\ell_i,(t,x,y)\big)\big\}\Big) = \begin{cases}\frac{1}{2}, & \text{if } y=2\\ 0, & \text{else}\end{cases}$ for $i \in \{3,4\}$
- $\mathcal{R}'\Big(\big(\ell_i,(t,x,y)\big),\big\{\big(\ell_i,(t,x,y)\big)\big\}\Big) = 1$ for $i \in \{1,2\}$
- $\mathcal{R}'\Big(a,A\Big) = 0$ for all other $(a,A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2, \lambda_2(t) = \frac{p_{\mathcal{N}_f}(t)}{\int_t^\infty p_{\mathcal{N}_f}(x)dx}$. Here, $p_{\mathcal{N}_f}$ is the PDF of a folded normal distribution $\mathcal{N}_f(\mu,\sigma^2)$. Furthermore, $S, \bar{S}$ are defined as in [15] and $\mathcal{Q}$ is defined in variant 1.

Variant 1 for case B in the formalism of Delicaris et al. Case A can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (\mathit{Loc}, \mathit{Var}_C, \mathit{Var}_R, \mathit{Distr}, \mathit{Inv}, \mathit{Init}, \mathit{Flow}_C, \mathit{Flow}_R, \mathit{Edge}_C, \mathit{Edge}_R)$ is a RAR, with

- $\mathit{Loc} = \{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4\}$ is a finite set of locations
- $\mathit{Var}_C = \{x, y\}$ is a finite ordered set of variables
- $\mathit{Var}_R = \{r_1, r_2\}$ is a finite ordered set of random clocks
- $\mathit{Distr} : \mathit{Var}_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $\mathit{Distr}(r_1) = \mathit{Exp}(\lambda_1)$
  - $\mathit{Distr}(r_2) = \mathit{Exp}(\lambda_2)$
- $\mathit{Inv} : \mathit{Loc} \to \mathbb{I}^2$ is a function assigning an invariant to each location:
  - $\mathit{Inv}(\ell_2) = \mathbb{R} \times (-\infty, 2]$
  - $\mathit{Inv}(\ell_i) = \mathbb{R}^2$ for $i \in \{0,1,3,4\}$
- $\mathit{Init} : \mathit{Loc} \to \mathbb{I}^2$ is a function assigning initial states to each location:
  - $\mathit{Init}(\ell_0) = [0,0]^2$
  - $\mathit{Init}(\ell_i) = \emptyset$ for $i \in \{1,2,3,4\}$
- $\mathit{Flow}_C : \mathit{Loc} \to \mathbb{I}^2$ is a function assigning a flow to each location:
  - $\mathit{Flow}_C(\ell_0) = ([2,2],[0,0])$
  - $\mathit{Flow}_C(\ell_2) = ([0,0],[1,1])$
  - $\mathit{Flow}_C(\ell_i) = ([0,0],[0,0])$ for $i \in \{1,3,4\}$
- $\mathit{Flow}_R : \mathit{Loc} \to \mathbb{I}^2$ is a function assigning a flow for the random clocks to each location:
  - $\mathit{Flow}_R(\ell_0) = ([1,1],[1,1])$
  - $\mathit{Flow}_R(\ell_i) = ([0,0],[0,0])$ for $i \in \{1,2,3,4\}$
- $\mathit{Edge}_C = \{e_3, e_4\} \subseteq \mathit{Loc} \times \mathbb{I}^2 \times \mathbb{I}^2 \times 2^{\mathit{Var}_C} \times \mathit{Loc}$ is a finite set of *non-stochastic jumps*:
  - $e_3 = (\ell_2, \mathbb{R} \times [2,2], \mathbb{R} \times [2,2], \emptyset, \ell_3)$
  - $e_4 = (\ell_2, \mathbb{R} \times [2,2], \mathbb{R} \times [2,2], \emptyset, \ell_4)$

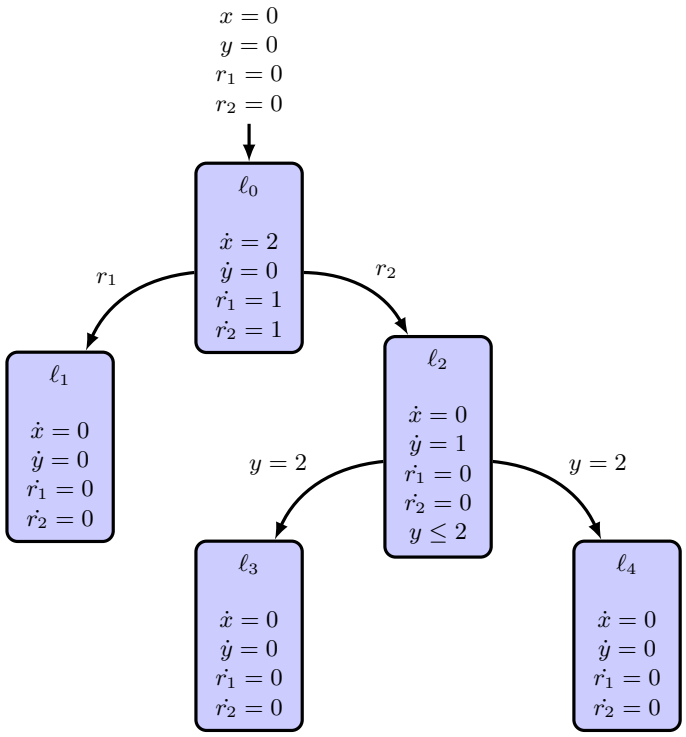

Figure 4: Case B modelled as a rectangular automaton with random clocks (RAR).

- $Edge_R = \{e_1, e_2\} \subseteq Loc \times Var_R \times Loc$ is a finite set of *stochastic jumps*:
  - $e_1 = (\ell_0, r_1, \ell_1)$
  - $e_2 = (\ell_0, r_2, \ell_2)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 4 for a graphical representation of this model as a RAR.

Variant 2 for case B in the formalism of Delicaris et al. Case B can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (Loc, Var_C, Var_R, Distr', Inv, Init, Flow_C, Flow_R, Edge_C, Edge_R)$ is a RAR, with

- $Distr' : Var_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $Distr'(r_1) = Exp(\lambda_1)$
  - $Distr'(r_2) = \mathcal{N}_f(\mu, \sigma^2)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2$, and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 4 for a graphical representation of this model as a RAR.

4.2.4 Case C

The general idea of case C is similar to case B. However, instead of a fixed time which is spent between the expiration of the second random variable and the discrete decision,

another continuous variable has to reach a specific value to trigger the nondeterministic choice. Additionally, the derivative of that continuous variable is given by a constant plus a normally distributed disturbance with mean μ and variance σ^2.

Again, when modelling this case with [15], the discrete nondeterminism is resolved probabilistically.

Variant 1 for case C in the formalism of Lygeros and Prandini Case C can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_B = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, \mathcal{R})$ is a SHA, with

- $\mathcal{Q} = \{\ell_0, \ell_1, \ell_2, \ell_3, \ell_4\}$ discrete states
- $\mathcal{X} = \mathbb{R}^3$ continuous states
- $Dom(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom(\ell_i) = \mathbb{R}^3$ for $i \in \{0, 1, 3, 4\}$
  - $Dom(\ell_2) = \mathbb{R} \times (-\infty, 2) \times \mathbb{R}$
- $f : S \to \mathbb{R}^3$ is a vector field:
  - $f(\ell_0, (x, y, z)) = (2, 0, 0)$
  - $f(\ell_1, (x, y, z)) = (0, 0, 0)$
  - $f(\ell_2, (x, y, z)) = (0, 1 + z, 0)$
  - $f(\ell_3, (x, y, z)) = (0, 0, 0)$
  - $f(\ell_4, (x, y, z)) = (-3, 0, 0)$
- $g : S \to \mathbb{R}^{3\times 1}$ is a diffusion coefficient:
  - $g(\ell_i, (x, y, z)) = (0, 0, 0)$ for $i \in \{0, 1, 2, 3, 4\}$
- $Init : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init(\ell_0, (0, 0, 0)) = 1$
  - $Init(a) = 0$ for any other $a \in S$
- $\lambda : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda(\ell_0, (x, y, z)) = \lambda_1 + \lambda_2$
  - $\lambda(\ell_i, (x, y, z)) = 0$ for $i \in \{1, 2, 3, 4\}$
- $\mathcal{R} : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:
  - $\mathcal{R}\Big(\big(\ell_0, (x, y, z)\big), \big\{\big(\ell_1, (x, y, z)\big)\big\}\Big) = \frac{\lambda_1}{\lambda_1 + \lambda_2}$
  - $\mathcal{R}\Big(\big(\ell_0, (x, y, z)\big), \big(\{\ell_2\} \times \{x\} \times \{y\} \times Z\big)\Big) = \frac{\lambda_2}{\lambda_1 + \lambda_2} \cdot \int_Z \frac{1}{\sigma \cdot \sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{z' - \mu}{\sigma}\right)^2} dz'$
  - $\mathcal{R}\Big(\big(\ell_2, (x, y, z)\big), \big\{\big(\ell_i, (x, y, z)\big)\big\}\Big) = \begin{cases} \frac{1}{2}, & \text{if } y = 2 \\ 0, & \text{else} \end{cases}$ for $i \in \{3, 4\}$
  - $\mathcal{R}\Big(\big(\ell_i, (x, y, z)\big), \big\{\big(\ell_i, (x, y, z)\big)\big\}\Big) = 1$ for $i \in \{1, 3, 4\}$
  - $\mathcal{R}\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $Z \in \mathcal{B}(\mathbb{R}), \lambda_1 = 0.1, \lambda_2 = 0.08, \mu = 0, \sigma^2 = 2$ and $S, \bar{S}$ as defined in [15].

Variant 2 for case C in the formalism of Lygeros and Prandini Case C can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_B = (\mathcal{Q}, \mathcal{X}', Dom', f', g', Init', \lambda', \mathcal{R}')$ is a SHA, with

- $\mathcal{X}' = \mathbb{R}^3$ continuous states
- $Dom'(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom'(\ell_i) = \mathbb{R}^4$ for $i \in \{0, 1, 3, 4\}$
  - $Dom'(\ell_2) = \mathbb{R}^2 \times (-\infty, 2) \times \mathbb{R}$
- $f' : S \to \mathbb{R}^4$ is a vector field:
  - $f'(\ell_0, (t, x, y, z)) = (1, 2, 0, 0)$
  - $f'(\ell_1, (t, x, y, z)) = (0, 0, 0, 0)$
  - $f'(\ell_2, (t, x, y, z)) = (0, 0, 1 + z, 0)$
  - $f'(\ell_3, (t, x, y, z)) = (0, 0, 0, 0)$
  - $f'(\ell_4, (t, x, y, z)) = (0, -3, 0, 0)$
- $g' : S \to \mathbb{R}^{4\times 1}$ is a diffusion coefficient: $g'(\ell_i, (t, x, y, z)) = (0, 0, 0, 0)$ for $i \in \{0, 1, 2, 3, 4\}$
- $Init' : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init'(\ell_0, (0, 0, 0, 0)) = 1$
  - $Init'(a) = 0$ for any other $a \in S$
- $\lambda' : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda'(\ell_0, (t, x)) = \lambda_1 + \lambda_2(t)$
  - $\lambda'(\ell_i, (t, x, y, z)) = 0$ for $i \in \{1, 2, 3, 4\}$
- $\mathcal{R}' : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:
  - $\mathcal{R}'\Big((\ell_0, (t, x, y)), \{(\ell_1, (t, x, y))\}\Big) = \frac{\lambda_1(t)}{\lambda_1(t)+\lambda_2(t)}$ for $\lambda_1(t) = \lambda_1$
  - $\mathcal{R}'\Big((\ell_0, (t, x, y, z)), (\{\ell_2\}\times\{t\}\times\{x\}\times\{y\}\times Z)\Big) = \frac{\lambda_2(t)}{\lambda_1(t)+\lambda_2(t)} \cdot \int_Z \frac{1}{\sigma\cdot\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{z'-\mu}{\sigma}\right)^2} dz'$ for $\lambda_1(t) = \lambda_1$
  - $\mathcal{R}'\Big((\ell_2, (t, x, y, z)), \{(\ell_i, (t, x, y, z))\}\Big) = \begin{cases} \frac{1}{2}, & \text{if } y = 2 \\ 0, & \text{else} \end{cases}$ for $i \in \{3, 4\}$
  - $\mathcal{R}'\Big((\ell_i, (t, x, y, z)), \{(\ell_i, (t, x, y, z))\}\Big) = 1$ for $i \in \{1, 2\}$
  - $\mathcal{R}'\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $Z \in \mathcal{B}(\mathbb{R}), \lambda_1 = 0.1, \mu = 5, \sigma = 2, \lambda_2(t) = \frac{p_{\mathcal{N}_f}(t)}{\int_t^\infty p_{\mathcal{N}_f}(x)dx}$. Here, $p_{\mathcal{N}_f}$ is the PDF of a folded normal distribution $\mathcal{N}_f(\mu, \sigma^2)$. Furthermore, $S, \bar{S}$ are defined as in [15] and $\mathcal{Q}$ is defined in variant 1.

Case C in the formalism of Delicaris et al. In the formalism of Delicaris et al. it is currently not possible to have derivatives which are influenced by stochastic noise. However, the normally distributed disturbance could be approximated by an appropriate specification of the interval describing the flows.

4.2.5 Case D

The general idea of case D is to model a continuous variable that initially increases and a race condition between two random variables, where the evolution of the continuous variable is stopped, if the first random variable wins the race; and the continuous variable is decreased, if the second random variable wins. Compared to case A, the domain of mode ℓ_0 is restricted which induces a forced leave of this mode. The goal is to find the probability that the continuous variable reaches a specific value < 0.

We consider two sets of random variables for all cases: 1) both random variables are distributed exponentially with rates λ_1 and λ_2, and 2) one variable is distributed exponentially with rate λ_1 and the other follows a folded normal distribution with parameters μ and σ.

Variant 1 for case D in the formalism of Lygeros and Prandini Case D can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_A = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda, \mathcal{R})$ is a SHA, with

- $\mathcal{Q} = \{\ell_0, \ell_1, \ell_2, \ell_3\}$ discrete states
- $\mathcal{X} = \mathbb{R}$ continuous states
- $Dom(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom(\ell_0) = (-\infty, 6)$
  - $Dom(\ell_i) = \mathbb{R}$ for $i \in \{1, 2, 3\}$
- $f : S \to \mathbb{R}$ is a vector field:
  - $f(\ell_0, x) = 2$
  - $f(\ell_1, x) = 0$
  - $f(\ell_2, x) = -3$
  - $f(\ell_3, x) = 0$
- $g : S \to \mathbb{R}^{1\times 1}$ is a diffusion coefficient: $g(\ell_i, x) = 0$ for $i \in \{0, 1, 2, 3\}$
- $Init : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init(\ell_0, 0) = 1$
  - $Init(a) = 0$ for any other $a \in S$
- $\lambda : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda(\ell_0, x) = \lambda_1 + \lambda_2$
  - $\lambda(\ell_i, x) = 0$ for $i \in \{1, 2, 3\}$
- $\mathcal{R} : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:

- $\mathcal{R}\Big(\big(\ell_0, x\big), \big\{\big(\ell_i, x\big)\big\}\Big) = \begin{cases} \frac{\lambda_i}{\lambda_1+\lambda_2}, & \text{if } x < 6 \\ 0, & \text{else} \end{cases}$ for $i \in \{1, 2\}$
- $\mathcal{R}\Big(\big(\ell_0, x\big), \big\{\big(\ell_3, x\big)\big\}\Big) = \begin{cases} 1, & \text{if } x = 6 \\ 0, & \text{else} \end{cases}$
- $\mathcal{R}\Big(\big(\ell_i, x\big), \big\{\big(\ell_i, x\big)\big\}\Big) = 1$ for $i \in \{1, 2, 3\}$
- $\mathcal{R}\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $S, \bar{S}$ as defined in [15].

Variant 2 for case D in the formalism of Lygeros and Prandini Case D can be modelled as a stochastic hybrid automaton following the syntax and semantics of [15], s.t. $H_A = (\mathcal{Q}, \mathcal{X}, Dom, f, g, Init, \lambda', \mathcal{R}')$ is a SHA, with

- $\mathcal{X}' = \mathbb{R}^2$ continuous states
- $Dom'(\ell) : \mathcal{Q} \to 2^{\mathcal{X}}$ is a set valued map assigning to each $\ell \in \mathcal{Q}$ an open subset of $\mathbb{R}$:
  - $Dom'(\ell_0) = \mathbb{R} \times (-\infty, 6)$
  - $Dom'(\ell_i) = \mathbb{R}^2$ for $i \in \{1, 2\}$
- $f' : S \to \mathbb{R}^3$ is a vector field:
  - $f'(\ell_0, (t, x)) = (1, 2)$
  - $f'(\ell_1, (t, x)) = (1, 0)$
  - $f'(\ell_2, (t, x)) = (1, -3)$
- $g' : S \to \mathbb{R}^{2\times 1}$ is a diffusion coefficient: $g'(\ell_i, (t, y)) = (0, 0)$ for $i \in \{0, 1, 2\}$
- $Init' : \mathcal{B}(S) \to [0, 1]$ is an initial probability measure on $(S, \mathcal{B}(S))$:
  - $Init'(\ell_0, (0, 0)) = 1$
  - $Init'(a) = 0$ for any other $a \in S$
- $\lambda' : S \to \mathbb{R}^+$ is a transition rate function:
  - $\lambda'(\ell_0, (t, x)) = \lambda_1 + \lambda_2(t)$
  - $\lambda'(\ell_i, x) = 0$ for $i \in \{1, 2\}$
- $\mathcal{R}' : \bar{S} \times \mathcal{B}(S) \to [0, 1]$ is a transition measure:
  - $\mathcal{R}'\Big(\big(\ell_0, (t, x)\big), \big\{\big(\ell_i, (t, x)\big)\big\}\Big) = \begin{cases} \frac{\lambda_i(t)}{\lambda_1(t)+\lambda_2(t)}, & \text{if } x < 6 \\ 0, & \text{else} \end{cases}$ for $i \in \{1, 2\}$ and $\lambda_1(t) = \lambda_1$
  - $\mathcal{R}'\Big(\big(\ell_0, (t, x)\big), \big\{\big(\ell_3, (t, x)\big)\big\}\Big) = \begin{cases} 1, & \text{if } x = 6 \\ 0, & \text{else} \end{cases}$
  - $\mathcal{R}'\Big(\big(\ell_i, (t, x)\big), \big\{\big(\ell_i, (t, x)\big)\big\}\Big) = 1$ for $i \in \{1, 2\}$

- $\mathcal{R}'\Big(a, A\Big) = 0$ for all other $(a, A) \in \bar{S} \times \mathcal{B}(S)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2, \lambda_2(t) = \frac{p_{\mathcal{N}_f}(t)}{\int_t^\infty p_{\mathcal{N}_f}(x)dx}$. Here, $p_{\mathcal{N}_f}$ is the PDF of a folded normal distribution $\mathcal{N}_f(\mu, \sigma^2)$. Furthermore, $S, \bar{S}$ are defined as in [15] and $\mathcal{Q}$ is defined in variant 1.

Variant 1 for case D in the formalism of Delicaris et al. Case D can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (Loc, Var_C, Var_R, Distr, Inv, Init, Flow_C, Flow_R, Edge_C, Edge_R)$ is a RAR, with

- $Loc = \{\ell_0, \ell_1, \ell_2, \ell_3\}$ is a finite set of locations
- $Var_C = \{x\}$ is a finite ordered set of variables
- $Var_R = \{r_1, r_2\}$ is a finite ordered set of random clocks
- $Distr : Var_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $Distr(r_1) = Exp(\lambda_1)$
  - $Distr(r_2) = Exp(\lambda_2)$
- $Inv : Loc \to \mathbb{I}^1$ is a function assigning an invariant to each location:
  - $Inv(\ell_0) = (-\infty, 2]$
  - $Inv(\ell_i) = \mathbb{R}$ for $i \in \{1, 2, 3\}$
- $Init : Loc \to \mathbb{I}^1$ is a function assigning initial states to each location:
  - $Init(\ell_0) = [0, 0]$
  - $Init(\ell_i) = \emptyset$ for $i \in \{1, 2, 3\}$
- $Flow_C : Loc \to \mathbb{I}^1$ is a function assigning a flow to each location:
  - $Flow_C(\ell_0) = [2, 2]$
  - $Flow_C(\ell_i) = [0, 0]$ for $i \in \{1, 3\}$
  - $Flow_C(\ell_2) = [-3, -3]$
- $Flow_R : Loc \to \mathbb{I}^2$ is a function assigning a flow for the random clocks to each location:
  - $Flow_R(\ell_0) = ([1, 1], [1, 1])$
  - $Flow_R(\ell_i) = ([0, 0], [0, 0])$ for $i \in \{1, 2, 3\}$
- $Edge_C = \{e_3\} \subseteq Loc \times \mathbb{I}^1 \times \mathbb{I}^1 \times 2^{Var_C} \times Loc$ is a finite set of *non-stochastic jumps*:
  - $e_3 = (\ell_2, [6, 6], [6, 6], \emptyset, \ell_3)$
- $Edge_R = \{e_1, e_2\} \subseteq Loc \times Var_R \times Loc$ is a finite set of *stochastic jumps*:
  - $e_1 = (\ell_0, r_1, \ell_1)$
  - $e_2 = (\ell_0, r_2, \ell_2)$

with $\lambda_1 = 0.1, \lambda_2 = 0.08$ and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 5 for a graphical representation of this model as a RAR.

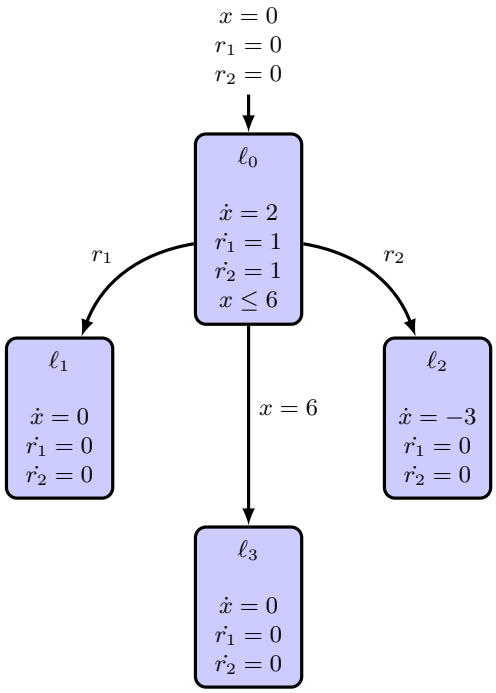

Figure 5: Case D modelled as a rectangular automaton with random clocks (RAR).

Variant 2 for case D in the formalism of Delicaris et al. Case D can be modelled as a rectangular automaton with random clocks (RAR) following the syntax and semantics of [3], s.t. $H_A = (Loc, Var_C, Var_R, Distr', Inv, Init, Flow_C, Flow_R, Edge_C, Edge_R)$ is a RAR, with

- $Distr' : Var_R \to \mathbb{F}_c$ is a function assigning a distribution to each random clock:
  - $Distr'(r_1) = Exp(\lambda_1)$
  - $Distr'(r_2) = \mathcal{N}_f(\mu, \sigma^2)$

with $\lambda_1 = 0.1, \mu = 5, \sigma = 2$, and $\mathbb{F}_c$ set of continuous distributions and $\mathbb{I}$ set of intervals as defined in [3]. Compare Figure 5 for a graphical representation of this model as a RAR.

5 Friendly Competition Results

5.1 Package Delivery (extended)

The tool SySCoRe has been applied to the updated version of the package delivery benchmark described in Sec. 4.1. The tool generates a satisfying controller in 15.5 seconds. Fig. 6 displays the obtained satisfaction probability as a function of the initial state.

5.2 Minimal examples results

For the computation of results for the minimal examples, two variants for the stochastic component are proposed. The parameters for those variants are as follows:

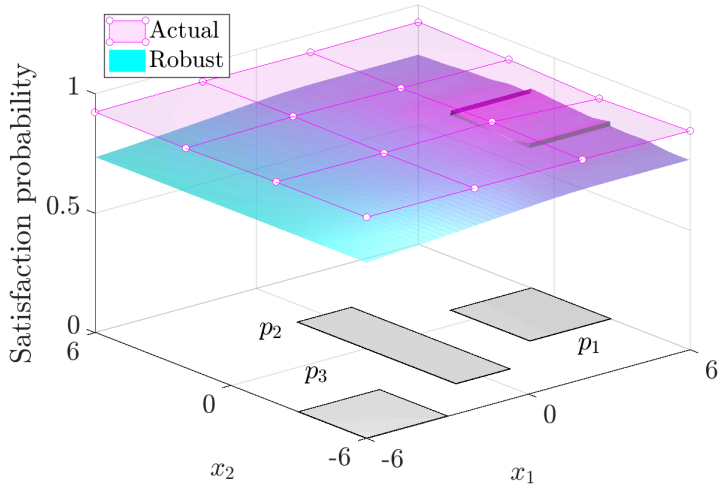

Figure 6: Comparison of the lower bound on the satisfaction probability obtained using SySCoRe (in cyan) and the actual satisfaction probability approximated using Monte Carlo simulation with a confidence of 90% (in magenta) as functions of the initial state for the extended package delivery study.

1. $\lambda_1 = 0.1, \lambda_2 = 0.08$,
2. $\lambda_1 = 0.1, \mu = 5, \sigma^2 = 2$.

If random variables are used to model the stochastic delays, in variant 1, one random variable that follows a exponential distribution with parameter $\lambda = \lambda_1 + \lambda_2 = 0.9$ is sufficient.

The results for HYPEG, ProbReach and RealySt are presented in Table 3 (variant 1) and Table 4 (variant 2). Note that no results for case C have been included, since all participating tools are not able to deal with stochastic noise in the evolution of continuous variables.

Each row in the table gives results for one tool with one specific method of execution. In column *Method*, the different modes are indicated with keywords, which are explained separately for every tool. Also, the handling of the nondeterminism in the model is indicated here: "**max**" and "**min**" refer to an optimization of the nondeterminism, i.e. probabilities are maximized resp. minimized; and "**prob**" means, that all nondeterminism is resolved probabilistically.

Computation To compute results with the tool HYPEG, the models are transformed into the formalism of *hybrid Petri nets with general transitions* [5]. In the default setting of HYPEG (indicated by "**SMC**"), nondeterminism is resolved uniformly, if existent (as in case B). By applying Q-learning (indicated by "**Q-learn**"), nonprophetic memoryless schedulers are trained to maximize or minimize the reachability probability. We performed 20000 training runs with a discretization truncating the continuous variables after the first decimal place, which is required and used for learning only. Afterwards, we used statistical model checking with the learned scheduler to estimate the probability. Since cases A and D do not exhibit nondeterminism, applying Q-learning results in the same method as the default setting (indicated by "**SMC**") and hence we omitted these computation results. In all cases, the confidence level was set to 95 % with a half interval width of 0.005.

Table 3: Results for minimal example cases A, B and D in variant 1. Results contain the computed probability, enclosures, error(s) or confidence interval if available, and computation times.

Tools		Case / Time bound			
Tool	Method	A / 10	B / 10	B / 12	D / 10
HYPEG	**SMC**, **prob**	0.289936 ±0.005 @ 95 % 0.268s	0.123475 ±0.005 @ 95 % 0.168s	0.146434 ±0.005 @ 95 % 0.213s	0.185348 ±0.005 @ 95 % 0.195s
	Q-learn, **max**		0.250857 ±0.005 @ 95 % 0.474s	0.289416 ±0.005 @ 95 % 0.547s	
	Q-learn, **min**		0.000000 ±0.005 @ 95 % 0.289s	0.000000 ±0.005 @ 95 % 0.313s	
ProbReach	**encl**, **max**	[0.28632,0.29105] 45.1s	[0.24884,0.25357] 39.5s	[0.28632,0.29105] 67.6s	[0.18456,0.18850] 118.2s
	CI	0.287113 ±0.005 @ 99 % 211.4s	0.249984 ±0.005 @ 99 % 172.9s	0.290366 ±0.005 @ 99 % 214.3s	0.185633 ±0.005 @ 99 % 155.6s
RealySt	**max**	0.288236 e_{stat}: $4.651 \cdot 10^{-4}$ e_∞: $3.808 \cdot 10^{-4}$ 0.0305276s	0.250016 e_{stat}: $3.239 \cdot 10^{-4}$ e_∞: $3.808 \cdot 10^{-4}$ 0.0649537s	0.288236 e_{stat}: $4.651 \cdot 10^{-4}$ e_∞: $3.808 \cdot 10^{-4}$ 0.0643815s	0.185280 e_{stat}: $2.061 \cdot 10^{-4}$ e_∞: $3.808 \cdot 10^{-4}$ 0.0305565s

ProbReach was used to compute rigorous enclosures (i.e., numerically sound intervals) that are guaranteed to contain the sought reachability probability (indicated by "**encl**"). Specifically, we asked for enclosures of width of no more than 0.005. Case B features a nondeterministic choice in location 2: for this model ProbReach computes an enclosure for the *maximum* reachability probability. Similarly, we computed confidence intervals (indicated by "**CI**") using the Bayesian estimation algorithm of ProbReach with the half width of the intervals at 0.005 and confidence level (in fact, posterior probability) at 99 %.

RealySt is specifically designed to resolve nondeterminism prophetically in *rectangular automata with random clocks (RAR)*, while also able to compute fully stochastic models. The minimal examples can all be modelled as singular automata with random clocks (a subclass of RAR), where case B exhibits discrete nondeterminism via a choice between two transitions and cases A and D are fully stochastic. While reachability is computed exactly in convex polytope representation, the integration method takes a number of integration samples as well as an integration bound as parameters. The number of integration samples affects the statistical error that is indicated by "e_{stat}" (more samples result in a smaller error). The integration bound limits the integration domain; all exceeding probability mass is cut off. The maximum cut off probability mass is given by "e_∞" (note that this is an overapproximation of the actual cut off probability mass). For the computations, we chose 100000 integration samples and an integration bound of 100.

Platforms Computation of results have been performed with different machines. HYPEG has been executed on a machine with an AMD Ryzen 7 PRO 5850U CPU and 32 GB of RAM.

Table 4: Results for minimal example cases A, B and D in variant 2. Results contain the computed probability, enclosures, error(s) or confidence interval if available, and computation times.

Tools		Case / Time bound			
Tool	Method	A / 10	B / 10	B / 12	D / 10
HYPEG	**SMC, prob**	0.446308 ±0.005 @ 95 % 0.373s	0.153738 ±0.005 @ 95 % 0.195s	0.225403 ±0.005 @ 95 % 0.298s	0.129474 ±0.005 @ 95 % 0.161s
	Q-learn, max		0.308441 ±0.005 @ 95 % 0.586s	0.448878 ±0.005 @ 95 % 0.669s	
	Q-learn, min		0.000000 ±0.005 @ 95 % 0.35s	0.000000 ±0.005 @ 95 % 0.334s	
ProbReach	**encl, max**	[0.44651,0.45027] 87.8s	[0.30758,0.31087] 82.4s	[0.44651,0.45027] 88.1s	[0.12907,0.13260] 41.8s
	CI	0.452306 ±0.005 @ 99 % 439.1s	0.310493 ±0.005 @ 99 % 232.7s	0.446794 ±0.005 @ 99 % 343.2s	0.131399 ±0.005 @ 99 % 121.1s
RealySt	**max**	0.448211 e_{stat}: $8.292 \cdot 10^{-5}$ e_{∞}: $2.061 \cdot 10^{-9}$ 0.02541s	0.308558 e_{stat}: $5.221 \cdot 10^{-5}$ e_{∞}: $2.061 \cdot 10^{-9}$ 0.0596879s	0.448211 e_{stat}: $8.292 \cdot 10^{-5}$ e_{∞}: $2.061 \cdot 10^{-9}$ 0.059773s	0.130280 e_{stat}: $1.766 \cdot 10^{-5}$ e_{∞}: $2.061 \cdot 10^{-9}$ 0.0320397s

ProbReach was run on a Linux virtual machine with an Intel Skylake CPU (six cores, 16MB cache, 2.6GHz) and 8GB of RAM; the computation time reported is the elapsed real (wall-clock) time using six cores. RealySt was executed on a machine with an AMD Ryzen 7 PRO 5850U CPU and 32 GB of RAM via WSL.

Discussion Generally, the results of the different tools align well for all cases. In the fully stochastic cases A and D, all tools (approximately) compute the same probability for both variants. In the presence of nondeterminism (case B), however, it can be obtained that naturally the results for different tools diverge. Here, all maximizing approaches agree on a probability of ≈ 0.25 (time bound of 10) resp. ≈ 0.28 (time bound of 12) for variant 1. The approaches that resolve nondeterminism probabilistically result in a smaller probability for the same variant (≈ 0.12 for a time bound of 10 and ≈ 0.14 for a time bound of 12, as expected). Analogously, this also holds for variant 2. Note that the results for case A match the results of case B with the time bound of 12: this is expected due to the time delay of two time units.

For a more detailed discussion of the results, in particular comparing computation times off the different tools, we refer to [1].

6 Conclusion

The evaluation of benchmarks this year featured six tools, among these a novel tool (SySCoRe). Following the initiative of last year, which was focusing on the development of a set of minimal

benchmarks, we observe that a substantial effort in this thread of research was done by the group members. This led to a refined version of the collection of minimal benchmarks with an extended description that features multiple formalisms. Furthermore, an updated version of last year's novel benchmark on the package delivery system was featured this year.

References

[1] Alessandro Abate et al. "ARCH-COMP22 Category Report: Stochastic Models". In: vol. 90. EasyChair, Dec. 2022, pp. 113–141. DOI: 10.29007/LSVC. URL: https://gitlab.com/goranf/ARCH-COMP.

[2] Julius Adelt et al. "Towards Safe and Resilient Hybrid Systems in the Presence of Learning and Uncertainty". In: *Leveraging Applications of Formal Methods, Verification and Validation. Verification Principles - 11th International Symposium, ISoLA 2022, Rhodes, Greece, October 22-30, 2022, Proceedings, Part I*. Ed. by Tiziana Margaria and Bernhard Steffen. Vol. 13701. Lecture Notes in Computer Science. Springer, 2022, pp. 299–319. DOI: 10.1007/978-3-031-19849-6_18.

[3] Joanna Delicaris et al. "Maximizing Reachability Probabilities in Rectangular Automata with Random Clocks". In: *Theoretical Aspects of Software Engineering - 17th International Symposium, TASE 2023, Bristol, United Kingdom, July 4-6, 2023, Proceedings*. LNCS. Accepted for publication. Springer, 2023.

[4] Brian Gough. *Gnu Scientific Library Reference Manual*. Network Theory Ltd., 2009. ISBN: 0954612078.

[5] Marco Gribaudo and Anne Remke. "Hybrid Petri Nets with General One-Shot Transitions". In: *Performance Evaluation* 105 (2016), pp. 22–50.

[6] Sofie Haesaert and Sadegh Soudjani. "Robust dynamic programming for temporal logic control of stochastic systems". In: *IEEE Transactions on Automatic Control* (2020).

[7] Sofie Haesaert, Sadegh Soudjani, and Alessandro Abate. "Verification of general Markov decision processes by approximate similarity relations and policy refinement". In: *SIAM Journal on Control and Optimization* 55.4 (2017), pp. 2333–2367.

[8] Birgit van Huijgevoort et al. "SySCoRe: Synthesis via Stochastic Coupling Relations". In: *Proceedings of the 26th ACM International Conference on Hybrid Systems: Computation and Control*. HSCC '23. San Antonio, TX, USA: Association for Computing Machinery, 2023. DOI: 10.1145/3575870.3587123.

[9] Jannik Hüls, Henner Niehaus, and Anne Remke. "Hpnmg: A C++ Tool for Model Checking Hybrid Petri Nets with General Transitions". In: *12th International NASA Formal Methods Symposium, NFM 2020*. Springer, 2020.

[10] Jannik Hüls and Anne Remke. "Model Checking HPnGs in Multiple Dimensions: Representing State Sets as Convex Polytopes". In: *19th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems, FORTE 2019*. Vol. 11535. LNCS. Cham: Springer, 2019, pp. 148–166.

[11] Jannik Hüls et al. "Analyzing Hybrid Petri nets with multiple stochastic firings using HyPro". In: *11th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2017*. ACM, 2018, pp. 178–185.

[12] Jannik Hüls et al. “State-Space Construction of Hybrid Petri Nets with Multiple Stochastic Firings”. In: *16th International Conference on Quantitative Evaluation of Systems, QEST 2019*. Vol. 11785. LNCS. Cham, Switzerland: Springer, 2019, pp. 182–199.

[13] Jannik Hüls et al. “State-Space Construction of Hybrid Petri Nets with Multiple Stochastic Firings”. In: *ACM Transactions on Modeling and Computer Simulation* 31.3 (2021), pp. 1–37.

[14] Abolfazl Lavaei et al. “Automated Verification and Synthesis of Stochastic Hybrid Systems: A Survey”. In: *Automatica* 146 (2022).

[15] John Lygeros and Maria Prandini. “Stochastic Hybrid Systems: A Powerful Framework for Complex, Large Scale Applications”. In: *European Journal of Control* 16.6 (2010), pp. 583–594. ISSN: 0947-3580. DOI: https://doi.org/10.3166/ejc.16.583-594.

[16] Geoffrey McLachlan and David Peel. *Finite mixture models*. Wiley, 2000. ISBN: 0471006262.

[17] Mathis Niehage, Arnd Hartmanns, and Anne Remke. “Learning optimal decisions for stochastic hybrid systems”. In: *MEMOCODE ’21: 19th ACM-IEEE International Conference on Formal Methods and Models for System Design, Virtual Event, China, November 20 - 22, 2021*. Ed. by S. Arun-Kumar et al. ACM, 2021, pp. 44–55. DOI: 10.1145/3487212.3487339.

[18] Mathis Niehage, Carina Pilch, and Anne Remke. “Simulating Hybrid Petri nets with general transitions and non-linear differential equations”. In: *13th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2020*. ACM, 2020.

[19] Mathis Niehage and Anne Remke. “Learning that Grid-Convenience Does Not Hurt Resilience in the Presence of Uncertainty”. In: *Formal Modeling and Analysis of Timed Systems - 20th International Conference, FORMATS 2022, Warsaw, Poland, September 13-15, 2022, Proceedings*. Ed. by Sergiy Bogomolov and David Parker. Vol. 13465. Lecture Notes in Computer Science. Springer, 2022, pp. 298–306. DOI: 10.1007/978-3-031-15839-1_17.

[20] Carina Pilch, Mathis Niehage, and Anne Remke. “HPnGs go Non-Linear: Statistical Dependability Evaluation of Battery-Powered Systems”. In: *Proceedings of the 26th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*. MASCOTS 2018. Milwaukee, WI, USA: IEEE, 2018, pp. 157–169.

[21] Carina Pilch and Anne Remke. “HYPEG: Statistical Model Checking for hybrid Petri nets: Tool Paper”. In: *Proceedings of the 11th EAI International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS 2017. Venice, Italy: ACM, 2017, pp. 186–191.

[22] Carina Pilch and Anne Remke. “Statistical Model Checking for hybrid Petri nets with multiple general transitions”. In: *Proceedings of the 47th IEEE/IFIP International Conference on Dependable Systems and Networks*. IEEE, 2017, pp. 475–486.

[23] Carina Pilch, Stefan Schupp, and Anne Remke. “Optimizing Reachability Probabilities for a Restricted Class of Stochastic Hybrid Automata via Flowpipe-Construction”. In: *18th Int. Conf. on Quantitative Evaluation of Systems, QEST 2021*. Vol. 12846. LNCS. Springer, Cham, 2021, pp. 435–456.

[24] Stefan Schupp et al. “HyPro: A C++ Library of State Set Representations for Hybrid Systems Reachability Analysis”. In: *NASA Formal Methods*. Ed. by Clark Barrett, Misty Davies, and Temesghen Kahsai. Vol. 10227. Cham: Springer, 2017, pp. 288–294.

[25] Fedor Shmarov and Paolo Zuliani. "Probabilistic Hybrid Systems Verification via SMT and Monte Carlo Techniques". In: *HVC*. Vol. 10028. LNCS. 2016, pp. 152–168. ISBN: 978-3-319-49052-6.

[26] Fedor Shmarov and Paolo Zuliani. "ProbReach: Verified Probabilistic δ-Reachability for Stochastic Hybrid Systems". In: *HSCC*. ACM, 2015, pp. 134–139.

[27] Fedor Shmarov and Paolo Zuliani. "SMT-Based Reasoning for Uncertain Hybrid Domains". In: *AAAI-16 Workhop on Planning for Hybrid Systems*. 2016, pp. 624–630.

[28] Jonas Stübbe and Anne Remke. "Monte-Carlo Integration on a Union of Polytopes". In: *19th Cologne-Twente Workshop on Graphs and Combinatorial Optimization, Garmisch-Patenkirchen, Germany, June 20-23, 2023*. Accepted for publication. 2023.