EPiC Series in Computing
Volume 90, 2022, Pages 113–141
Proceedings of 9th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH22)

ARCH-COMP22 Category Report: Stochastic Models

Alessandro Abate[1], Henk Blom[2], Joanna Delicaris[3], Sofie Haesaert[4], Arnd Hartmanns[5], Birgit van Huijgevoort[4], Abolfazl Lavaei[6], Hao Ma[2], Mathis Niehage[3], Anne Remke[3], Oliver Schön[6], Stefan Schupp[7], Sadegh Soudjani[6], and Lisa Willemsen[3]

[1] University of Oxford, Oxford, UK
[2] Delft University of Technology, Delft, The Netherlands
[3] University of Münster, Münster, Germany
[4] Eindhoven University of Technology, Eindhoven, The Netherlands
[5] University of Twente, Enschede, The Netherlands
[6] Newcastle University, Newcastle upon Tyne, UK
[7] TU Wien, Vienna, Austria

Abstract

This report presents the results of a friendly competition for formal verification and policy synthesis of stochastic models. It also introduces new benchmarks and their properties within this category and recommends next steps for this category towards next year's edition of the competition. In comparison with tools on non-probabilistic models, the tools for stochastic models are at the early stages of development that do not allow full competition on a standard set of benchmarks. We report on an initiative to collect a set of minimal benchmarks that all such tools can run, thus facilitating the comparison between efficiency of the implemented techniques. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in Summer 2022.

1 Introduction

The subgroup "Stochastic Models" of the annual friendly ARCH-Competition focusses on recent developments of tools that can analyze systems which exhibit uncertain, stochastic behavior in its various expressions (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 modelling 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), 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 2022 in the group *stochastic models*. We refer the reader to the survey paper [53] 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): FIGARO workbench, HYPEG, the Modest Toolset, PyCATSHOO, RealySt, SDCPN & IPS, SySCoRe. In particular, the new tool RealySt joined the competition this year. A substantial new development was also observed in SySCoRe. The benchmark collection has been extended by two interesting benchmarks: a *Package Delivery* benchmark that include simple (linear) dynamics of the system but allow for defining more expressive specifications to go beyond safety and reachability. This benchmark can be used to check specifications that are expressed by various classes of finite state automata. We have also developed a new benchmark as a *minimal example* 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. Section 3 presents already established benchmarks and a set of new benchmark descriptions are presented in Section 4, which include a discussion of the individual models syntax and semantics. Next, 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

Here we present the tools which participated this year in alphabetical order.

FIGARO workbench The Figaro language, created in 1990, is a (free and public) domain specific object oriented modeling language dedicated to dependability. It generalizes all the usual reliability models, and can easily be associated to various graphical representations. It allows to cast generic models in knowledge bases (KB). A formal definition of its semantics is available in [11] and [12]. The Figaro workbench, mostly developed by EDF (Electricité de France) since the creation of the Figaro language, comprises a set of tools to create Figaro models and to process them in order to perform dependability analyses; the main tools are:

- FigaroIDE is an integrated development environment for creating KBs;
- KB3 is a generic graphical user interface. Once a KB has been loaded in KB3, it becomes a specialized GUI for building a certain kind of graphical models. KB3 comes with a few “abstract KBs” corresponding to classical reliability models, including reliability block diagrams, digraphs, Petri nets and BDMP (Boolean logic Driven Markov Processes). KB3

provides sophisticated functions to input and manage complex system models, perform interactive simulations and can generate fault trees and display them graphically. KB3 is intensively used at EDF to automatically generate fault trees and a few dynamic models for Probabilistic Safety Assessment of its nuclear power plants;

- Figseq [9, 14] is a quantification tool for continuous time Markov chains that explores the sequences leading to a target state, defined by a Boolean expression. Given the mission time and truncation criteria, Figseq computes an estimated value and an upper bound of the undesirable event probability. It can perform reliability and availability calculations;

- YAMS [10] is another solver: it uses Monte Carlo simulation on the system model to compute various quantities, including reliability and availability. Any kind of probability distribution can associated to transitions with this tool. YAMS is also able to output a selection of simulated scenarios, but the obtained results are much more "noisy" than those obtained with Figseq.

- All the above cited tools are "industry proof" tools, used in real studies of complex systems such as nuclear power plants, telecommunication and electrical networks... KB3 is commercially available under the name RiskSpectrum ModelBuilder. A new tool, still a prototype, is available to process FIGARO Markovian models: it is based on the STORM probabilistic model checker, cf [50, 49]. This open source tool, called STORM-Figaro, is now available on github. The paper [13] explains its principles, where to find it, how to use it and compares its performances to those of Figseq and YAMS on a set of test cases.

HYPEG The Java-based library HYPEG [66] implements time-bounded discrete-event simulation for hybrid Petri nets with general transitions (HPnGs) [37], 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 [67]. 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 [65, 64], whereas piecewise-linear continuous behavior is simulated without approximation. Recently, HYPEG was extended to resolve discrete nondeterminism using reinforcement learning to maximize or minimize the probability of a property [63].

The tool is available at `https://zivgitlab.uni-muenster.de/ag-sks/tools/HYPEG`.

The Modest Toolset A collection of tools for the modelling and analysis of stochastic timed and hybrid systems, ranging from discrete-time Markov chain models to stochastic hybrid systems with general probability distributions and nonlinear continuous dynamics, the Modest Toolset [43] notably includes the modes statistical model checker [16] and the prohver [42] safety checking tool that participate in the ARCH friendly competition. The Modest Toolset is currently jointly developed at the University of Twente and Saarland University. It is available online at modestchecker.net. All tools in the Modest Toolset support the formal Modest [42] and JANI [17] input languages; in this competition, we use Modest models handwritten to represent the respective benchmarks.

modes is, at its core, a Monte Carlo simulator. It implements an automated importance splitting method [15] to tackle the rare events problem, and lightweight scheduler sampling (LSS) [57] to find near-optimal decisions for nondeterministic choices. It contains simulation engines specialised to different semantic formalisms such as discrete-time Markov chains or probabilistic timed automata [51]. For this competition, its engines for singular and general non-linear stochastic hybrid automata are relevant; we use the former unless noted otherwise. In past ARCH competitions, modes exercised its rare event simulation capabilities; this time, it will be applied to nondeterministic models instead, exploiting its implementation of LSS. By sampling schedulers, LSS delivers under-/overapproximations of maximum/minimum reachability probabilities, respectively.

prohver model-checks safety properties of stochastic hybrid automata (SHA) [34] by (1) overapproximating continuous stochastic choices by splitting them into discrete probabilistic choices plus continuous nondeterminism, and (2) separating the numeric analysis into a non-stochastic computation of the reachable state set using a modified version of PHAVer [35] followed by reintroducing the probabilities to finally compute the value of interest on a Markov decision process. In this way, prohver computes an overapproximation of maximum reachability probabilities. In certain settings, such as singular probabilistic hybrid automata, where PHAVer is exact, prohver can also compute an exact (up to floating-point and value iteration errors) result. For models that sample from continuous probability distributions, prohver needs a user-specified set of intervals to split the distributions into. For this competition, we added a new algorithm that iteratively refines the provided intervals by dividing those intervals in two or more equal parts that led to the largest change in the resulting probability in the previous refinement. Still, as we saw in our experiments, the effectiveness of this method depends on a good choice of initial intervals.

PyCATSHOO The PyCATSHOO [26] tool has been developed in the R&D division of EDF. This development was motivated by the need to address, in some safety studies, the continuous deterministic phenomena that unfold in the studied systems. It provides modelling tools that take into account the synchronization between, on the one hand, the discrete stochastic behavior, classically taken into account in dependability-oriented modelling and, on the other hand, the 0D/1D physical modelling. PyCATSHOO is based on the theoretical framework of piecewise deterministic Markov processes (PDMP). It implements this theoretical framework through Distributed Stochastic Hybrid Automata (DSHA). PyCATSHOO leverages Hybrid Stochastic Automata (HSA) to implement PDMPs and introduced the notion of distribution which allows modular modelling and avoids the problem of the combinatorial explosion which SHAs suffer from when it comes to an industrial-sized system modelling. In a nutshell, PyCATSHOO is a dynamic library written in C++ that can be used via a C++ or a Python API. Thanks to a mainly declarative approach, this library allows the modelling of the discrete stochastic behavior of complex system actors. It also allows for an effective formulation of ordinary or algebraic differential equations that govern the continuous state variables of these actors. These equations are solved by PyCATSHOO and can be efficiently adapted to the different system's operating modes. Indeed, PyCATSHOO takes over boundary crossings and managing of a system multimodal behavior. Reconfigurations can thus be easily modelled, whether they are due to a deterministic behavior of the I&C or to stochastic events such as failures and repairs. PyCATSHOO embeds a Monte Carlo simulation engine where the development of an Importance Sampling algorithm is in process [25]. PyCATSHOO also provides a fault tree generator that can be used when a static view of the modelled system is required. Its open software architecture allows it to interoperate easily with other tools. PyCATSHOO can also be used to build generic modelers. This functionality has been used to develop the PyCABIA modeler which implements an extension of the reliability diagrams formalism. PyCABIA can be used in a static way and automatically generate fault trees. It can also be used in dynamic modelling. It then provides the notion of passive redundancies, shared resources, etc.

RealySt RealySt is a tool which optimizes reachability probabilities for the class of rectangular automata with random clocks, which exhibit discrete and continuous nondeterminism as well as stochasticity. Using forwards reachability anaylsis and a backwards refinement approach, probabilities can be optimized. It is implemented in C++ and relies on the library HyPro [68] for the state-set representation via convex polytopes as well as efficient geometric operations. The GNU Scientific Library (GSL) [36], providing Monte Carlo integration algorithms, is used for multi-dimensional integration.

RealySt builds on the tool hpnmg [45], a model checker for Hybrid Petri nets with an arbitrary but finite number of general transition firings against specifications formulated in STL [47]. Each general transition firing results in a random variable which follows a continuous probability distribution. It efficiently implements and combines algorithms for a symbolic state-space creation [46, 44], transformation to a geometric representation as convex polytopes [48], model checking a potentially nested STL formula and integrating over the resulting satisfaction set to yield the probability that the specification holds at a specific time.

RealySt is currently being developed within the DFG project 471367371 as a cooperation between the RWTH Aachen and the University of Münster.

RealySt is available at https://zivgitlab.uni-muenster.de/ag-sks/tools/realyst.

SySCoRe SySCoRe stands for Synthesis via Stochastic Coupling Relations for stochastic continuous state systems. This tool is developed for temporal logic control synthesis of discrete-time stochastic dynamical systems with outputs. It allows both model order reduction and space discretization while quantifying the error induced in the probability of satisfying the given property. The development of SySCoRe is based on the papers [38, 39, 40, 41, 77] and encode directly the coupling between stochastic processes into the simulation relation that assess the similarity between the associated dynamical systems. The developed algorithms compute two precision parameters (ϵ, δ), which allow bounding the deviations between models in both the output trajectories ϵ and the transition probabilities δ. The obtained abstract models, either with deterministic continuous states or with stochastic finite states, are then employed in probabilistic model checking. The current version of SySCoRe is capable of handling co-safe LTL properties with infinite horizon. 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 for infinite horizon properties. Besides that, it can handle an unbounded (e.g. Gaussian) additive disturbance.

It is worth noting that AMYTISS [52], StocHy [21], and FAUST^2 [72] did not participate in this year competition, since they do not natively support stochastic *hybrid* systems, which are the main benchmark models employed in this report. In particular, these are software tools for designing correct-by-construction controllers of stochastic discrete-time *control* systems. The underlying idea of the implemented algorithms is abstraction to finite Markov decision processes (MDPs) with error bounds formulated in a series of previous works [69, 70, 71]. The underlying computation part in AMYTISS is similar to the one used in FAUST^2, however, it is developed to solve a two-player stochastic game by providing parallel algorithms over GPUs and hardware accelerators [54, 55, 56]. StocHy, instead, leverages scalable and robust abstractions as interval MDPs [22, 21].

2.1 Frameworks

In contrast to complete tools, frameworks usually provide a collection of algorithms and data structures or collect several tools for different sub-problems into one library.

SDCPN & IPS is a reach probability modelling and estimation framework that has been developed for the evaluation of multi-actor air traffic designs on mid-air collision risk. Because this air traffic application domain is very demanding, the selected mathematical setting is General Stochastic Hybrid System (GSHS) [19]. GSHS incorporates Brownian motion in continuous-time Piecewise Deterministic Markov Processes [29]. Because a direct specification of a large GSHS model does not work, the framework of Stochastically and Dynamically Coloured Petri Nets (SDCPN) [30, 31, 32, 33] has been developed for the compositional specification of a GSHS model. For the acceleration of MC simulation of rare events, the Interacting Particle Systems (IPS) approach for GSHS is used [23, 7, 8, 58, 59]. The SDCPN & IPS framework is applied to the Heated Tank benchmark.

3 Established benchmarks, revisited

3.1 Building Automation Systems

The building automation benchmark is split into a 4 and 7-dimensional models with the aim of generating a control policy which maximises a safety problem. An in-depth description of the benchmark can be found in ARCH 2018 [4] and [20].

3.2 Heated Tank

The Heated Tank benchmark stems from the safety literature; there it is a well-known example of a Piecewise Deterministic Markov Process (PDMP) [29]. This made the Heated Tank benchmark a logical candidate for inclusion in the set of ARCH stochastic models [1, 2, 3, 4].

The heated tank system consists of a tank containing liquid whose level is influenced by two pumps and one valve managed by a controller. The purpose of the liquid in the tank is to absorb and transport energy from a heat source; this means that under nominal conditions one of the pumps produces a constant inflow of cool liquid while a similar flow of heated liquid leaves the tank through the valve. The Euclidean valued state components are height $x_{H,t}$ and temperature $x_{T,t}$ of the liquid in the tank at moment t. Pumps and Valve may fail, and a Controller switches Pumps or Valve if the height of the liquid becomes too high or too low. The reach probabilities to be estimated on a given time interval are: Dryout probability, Overflow probability, and Overheating probability.

In literature, e.g. [27, 75], the heated tank benchmark has five versions. In version 1, Pumps and Valve have constant failure rates. In version 2, Pumps and Valve have mode dependent failure rates. In version 3, the Controller in version 1 may forget to implement its switching decision. In version 4, the Pumps and Valve in version 1 are repaired. In version 5, the failure rates in version 1 depend on the liquid temperature. Because version 4 involves repairs of failed pumps and valve, its Dryout probability is much lower than for the other versions. Therefore in [4], version 4 has been selected as most suitable rare event estimation benchmark. In [2] relevant rare event extensions of this version have been identified. Table 1 gives an overview of these combinations, including the version number used within ARCH, and the relation to the version numbers in literature.

Table 1: Heated Tank benchmarks defined in [2].

ARCH version	4.0	4.I	4.II	4.III	4.IV	4.V
Based on version(s) in literature	4	2+4	3+4	5+4	4	4
Pumps and Valve failure	Y	Y	Y	Y	Y	Y
Pumps and Valve repair	Y	Y	Y	Y	Y	Y
Mode dependent failure rate	-	Y	-	-	-	-
Communication failure	-	-	Y	-	-	-
Temperature dependent failure rate	-	-	-	Y	-	-
Non-exponential failure / repair rate	-	-	-	-	Y	-
Brownian motion in Heat source	-	-	-	-	-	Y

In [2] version 4.0 has formally been described in the model specification language SDCPN, and in the languages Modest and HPnGs that are used by modes and HYPEG respectively. In [4, 2, 3], Heated Tank version 4.0 has been evaluated by the tools modes and HYPEG and by the framework SDCPN&IPS. In [1], Heated Tank version 4.III has been evaluated by the tools FIGARO and PyCATSHOO, and by the framework SDCPN&IPS.

3.3 Stochastic Van der Pol Oscillator

The discrete-time state evolution of the oscillator is given by:

$$
\begin{aligned}
x_1(k+1) &= x_1(k) + x_2(k)\tau + w_1(k) \\
x_2(k+1) &= x_2(k) + (-x_1(k) + (1 - x_1(k)^2)x_2(k))\tau + w_2(k),
\end{aligned}
\tag{1}
$$

where the sampling time τ is set to $0.1s$ and $(w_1(k), w_2(k))$ is a pair of stochastic noise signals at time k drawn from a uniform density function with a compact support $D = [-0.02, 0.02] \times [-0.02, 0.02]$.

Consider a safety specification for staying within the working area $A := [-5, 5] \times [-5, 5]$. This property is denoted by $\Box A$, where $\Box$ should be read as 'always'. Consider also the Büchi specification $\Box\Diamond B$, which means repeatedly reaching the target set $B := [-1.2, -0.9] \times [-2.9, -2]$. The notation $\Box\Diamond$ should be read as 'always eventually'. This property means the set B should be always visited in the future of the trajectory, and equivalently requires visiting B infinite number of times along a trajectory.

Problem 1 (Qualitative Verification)**.** *Compute the set of initial states from which the probability of satisfying the specification $\Box A \wedge \Box\Diamond B$ under dynamics (1) is equal to 1.*

Problem 2 (Quantitative Verification)**.** *Compute the probability of satisfying the specification $\Box A \wedge \Box\Diamond B$ under dynamics (1) as a function of initial state.*

Since some of the tools are not able to handle $\Box\Diamond B$, the following modified dynamical system can be used together with a reachability specification that gives an upper-bound for probability of satisfying $\Box A \wedge \Box\Diamond B$. Let us denote the right-hand side of (1) by $f(x(k)) + w(k)$. Define a new dynamical system with state space $A \cup \{\phi_1, \phi_2\}$ such that ϕ_1 and ϕ_2 are sink states and

$$x(k+1) = \begin{cases} f(x(k)) + w(k) & \text{if } w(k) \in A \backslash f(x(k)) \text{ and } x(k) \notin B \\ \phi_1 & \text{if } w(k) \notin A \backslash f(x(k)) \text{ and } x(k) \notin B \\ f(x(k)) + w(k) & \text{if } w(k) \in A \backslash f(x(k)) \text{ and } x(k) \in B \text{ and } \nu(k) = 0 \\ \phi_1 & \text{if } w(k) \notin A \backslash f(x(k)) \text{ and } x(k) \in B \text{ and } \nu(k) = 0 \\ \phi_2 & \text{if } w(k) \in A \backslash f(x(k)) \text{ and } x(k) \in B \text{ and } \nu(k) = 1 \\ \phi_2 & \text{if } w(k) \notin A \backslash f(x(k)) \text{ and } x(k) \in B \text{ and } \nu(k) = 1, \end{cases} \tag{2}$$

where $\nu(k)$ are independent and identically distributed Bernoulli random variables with success probability $(1 - \zeta)$.

Problem 3 (Quantitative Reachability)**.** *Compute the probability* $\Diamond \phi_2$ *under dynamics* (2)*.*

The solution of Problem 3 is an upper bound for Problem 2. Moreover, it converges to the solution of Problem 2 when $\zeta \to 1^-$.

The dynamics in (1) can be extended to include inputs for shaping the limiting behaviour of the system. Consider the non-autonomous version of the oscillator dynamics given by:

$$\begin{aligned} x_1(k+1) &= x_1(k) + x_2(k)\tau + w_1(k) \\ x_2(k+1) &= x_2(k) + (-x_1(k) + (1 - x_1(k)^2)x_2(k))\tau + u(k)w_2(k). \end{aligned} \tag{3}$$

Problem 4 (Quantitative Synthesis)**.** *Compute a policy for dynamical system* (3) *that maximises the probability of satisfying* $\Box A \wedge \Box \Diamond B$*.*

Instead of multiplicative noise it is also interesting to consider additive Gaussian noise. To this end, consider the following version of the oscillator dynamics

$$\begin{aligned} x_1(k+1) &= x_1(k) + x_2(k)\tau + w_1(k) \\ x_2(k+1) &= x_2(k) + (-x_1(k) + (1 - x_1(k)^2)x_2(k))\tau + u(k) + w_2(k), \end{aligned} \tag{4}$$

with $w \sim \mathcal{N}(0, 0.2I_2)$, where I_2 denotes the two-dimensional identity matrix.

Problem 5 (Quantitative Synthesis)**.** *Compute a policy for dynamical system* (4) *that maximises the probability of satisfying the scLTL specification* $A \cup B$*.*

4 New Benchmarks

In this section we present novel benchmarks or variants of old benchmarks that have been proposed during the competition which allow for new outcomes.

4.1 Package Delivery

With this case study we aim at showing if the tools can also be used to synthesize controllers for more complex specifications, i.e., for non-acyclic DFAs. For this, we consider the following setup:

Consider a simple time-discrete system with a continuous state x, control input u, and disturbance w. Assume that the system's dynamics are captured by the following equations:

$$\underbrace{\begin{bmatrix}\dot{x}_1\\ \dot{x}_2\end{bmatrix}}_{\dot{x}(t)} = A\underbrace{\begin{bmatrix}x_1\\ x_2\end{bmatrix}}_{x(t)} + B\underbrace{\begin{bmatrix}u_1\\ u_2\end{bmatrix}}_{u(t)} + \underbrace{\begin{bmatrix}w_1\\ w_2\end{bmatrix}}_{w(t)}, \tag{5}$$

$$y(t) = x(t), \tag{6}$$

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

$$A = \begin{bmatrix}0.9 & 0\\ 0 & 0.8\end{bmatrix} \quad B = \begin{bmatrix}1.4 & 0\\ 0 & 1.4\end{bmatrix},$$

and the noise w is normally distributed with mean $\begin{bmatrix}0; 0\end{bmatrix}$ and variance $0.2I_2$, that is $w \sim \mathcal{N}(0, 0.2I_2)$. These equations capture the dynamics of the agent in a package delivery scenario. For this, we define three regions p_1, p_2, and p_3 as follows: $p_1 := [5, 6]\times[-1, 1]$, $p_2 := [0, 1]\times[-5, 1]$ and $p_3 := [-4, -2] \times [-4, -3]$. The scenario is as follows: the agent can pick up a parcel at p_1 and must deliver it to p_3 (c.f. Fig. 1). If the agent visits p_2 while carrying a package, he loses the parcel and has to restart by picking up a new parcel at p_1. This corresponds to the *scLTL* specification $\Diamond(p_1 \wedge (\neg p_2 \,\mathsf{U}\, p_3))$ whose DFA is given in Fig. 2.

Problem 6. *Compute a policy for dynamical system* (5) *that maximises the probability of satisfying the* scLTL *specification* $\Diamond(p_1 \wedge (\neg p_2 \,\mathsf{U}\, p_3))$.

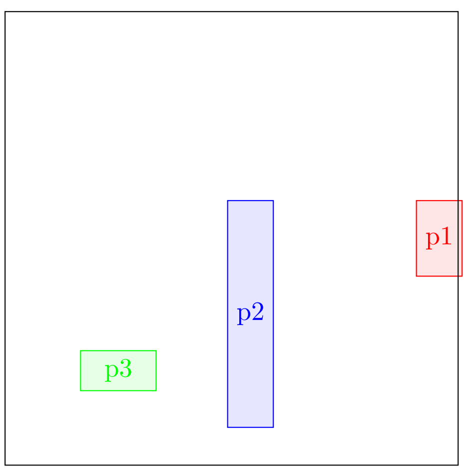

Figure 1: Regions defined over the output space

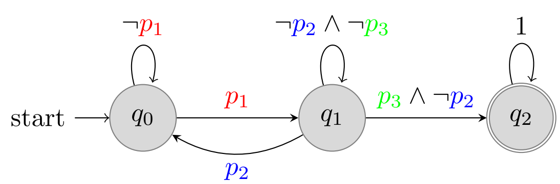

Figure 2: DFA

4.2 Van der Pol Oscillator in Continuous time

We define a specification on this system that is suitable for rare event estimation. We consider three polytopes

$$\mathcal{P}_i = \{X \in \mathbb{R}^2 \,|\, AX \leq B_i\}, \quad i \in \{\text{out}, \text{mid}, \text{in}\},$$

$\mathcal{P}_{\text{out}}$	(3.2351,-0.8270)	(1.1919,4.6216)	(-3.2351,0.8270)	(-1.1919, -4.6216)
$\mathcal{P}_{\text{in}}$	(1.3338,0.6432)	(1.0500,1.4000)	(-1.3338,-0.6432)	(-1.0500,-1.4000)
$\mathcal{P}_{\text{mid}}$	(2.2845,-0.0919)	(1.1209,3.0108)	(-2.2845,0.0919)	(-1.1209,-3.0108)

Table 2: Vertices of the polytopes in the specification (8) of the Van der Pol Oscillator for rare event estimation.

in the two-dimensional space. The two polytopes $\mathcal{P}_{\text{out}}$ and $\mathcal{P}_{\text{in}}$ specify a region around the limit cycle of the system. The $\mathcal{P}_{\text{in}}$ polytope specifies a area between the previous two polytopes. The dynamics of the system in continuous time is as follows

$$\begin{aligned} dx_1 &= x_2 dt + \sigma_1 dW_1 \\ dx_2 &= (-x_1 + (1 - x_1^2)x_2)dt + \sigma_2 dW_2, \end{aligned} \tag{7}$$

with W_1 and W_2 being independent standard Brownian motion.

Problem 7 (Rare event computation)**.** *Compute the probability that the trajectory goes outside of the region between $\mathcal{P}_{out}$ and $\mathcal{P}_{in}$ around the limit cycle in the time internal $[0, T]$ after entering the polytope $\mathcal{P}_{mid}$:*

$$\begin{aligned} &t_1 := \inf\{t; X_t \in \mathcal{P}_{mid}\} \\ &t_2 := \inf\{t > t_1; (X_t \in \mathcal{P}_{in}) \vee (X_t \in \mathbb{R}^2/\mathcal{P}_{out})\} \\ &\textit{compute or estimate } \ \mathbb{P}\{t_2 \leq T\}. \end{aligned} \tag{8}$$

The following numerical values can be used: time horizon $T = 13$, initial state $X_0 = [4, 2]^T$,

$$A = \begin{bmatrix} +\alpha_1 & -1 \\ -\alpha_2 & +1 \\ -\alpha_1 & +1 \\ +\alpha_2 & -1 \end{bmatrix}, \quad B_{\text{out}} = \begin{bmatrix} -\beta_1 \\ +\beta_2 \\ -\beta_1 \\ +\beta_2 \end{bmatrix}, \quad B_{\text{in}} = \begin{bmatrix} -\gamma_1 \\ +\gamma_2 \\ -\gamma_1 \\ +\gamma_2 \end{bmatrix}, \quad B_{\text{mid}} = (B_{\text{out}} + B_{\text{in}})/2,$$

where $\alpha_1 = 6/7, \alpha_2 = -8/3, \beta_1 = -2.9, \beta_2 = 7.2, \gamma_1 = -1, \gamma_2 = 4.5$. Sample trajectories of the system is plotted in Figure 3 together with polytopes $\mathcal{P}_{\text{out}}$ (in black), $\mathcal{P}_{\text{in}}$ (in green), $\mathcal{P}_{\text{mid}}$ (in blue), and the limit cycle for the deterministic version of the system (in blue). The diffusion terms are $\sigma_1 = \sigma_2 = 0.2$. Applying a standard Monte Carlo approach to this problem with $10,000$ trajectories gives the estimate 0.027 for the probability in (8). To reduce this probability further, we can change the values of β_i and γ_i, which are intercepts of the lines in the polytopes, to enlarge the region around the limit cycle (the region between the two polytopes). For this purpose, we choose $\beta_1 = -3.6$, $\beta_2 = 7.8$, $\gamma_1 = -0.5$, and $\gamma_2 = 4.2$. Table 2 summarises the vertices of the polytopes.

4.3 Comparison over 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. Case A (see 4.3.1) is the most simple one which contains two random variables modeling random transition delays. The corresponding property checks whether one of the other locations is

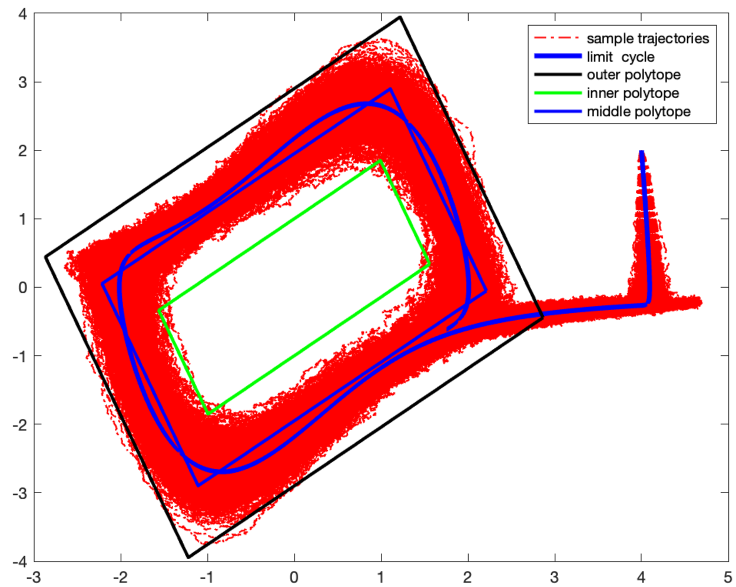

Figure 3: Sample trajectories of the stochastic Van der Pol Oscillator starting from initial state $X_0 = [4, 2]^T$. Most of the trajectories remain between the outer polytope $\mathcal{P}_{\text{out}}$ (black) and the inner polytope $\mathcal{P}_{\text{in}}$ (green).

reached before a specific time. Case B (see 4.3.2) extends case A by a **conflict** between two urgent discrete transitions. We consider different ways to resolve this discrete nondeterminism. Case C (see 4.3.3) is based on case B and adds **stochastic noise** to the automaton. Case D (see 4.3.4) is again an adaption of case A, where the time spent in the initial location is restricted by an invariant. This leads to a potential **time lock**, which different tools and approaches tackle differently.

All cases contain two random variables X_1 and X_2, which model the initial conflict between two transitions as a race condition. Initially, these random variables are exponentially distributed, however, different tools are also able to compute results for different distributions.

4.3.1 Case A

The automaton of this minimal example contains three locations and one continuous variable x. From the initial location ℓ_0, two transitions are enabled, which both correspond to a random variable. Hence, the time delay in location ℓ_0 depends on these random variables. The transition to location ℓ_1 corresponds to random variable X_1. Location ℓ_2 is reached with the expiration of another random variable X_2. An illustration is given in Figure 4.

To compare probabilities, we define property ϕ, which checks if the valuation of the continuous variable x reaches -1 before a total time of 10:

$$\phi = F^{\leq 10}(x \leq -1).$$

Due to the derivatives of x in the three locations, x can only reach the specified value, if

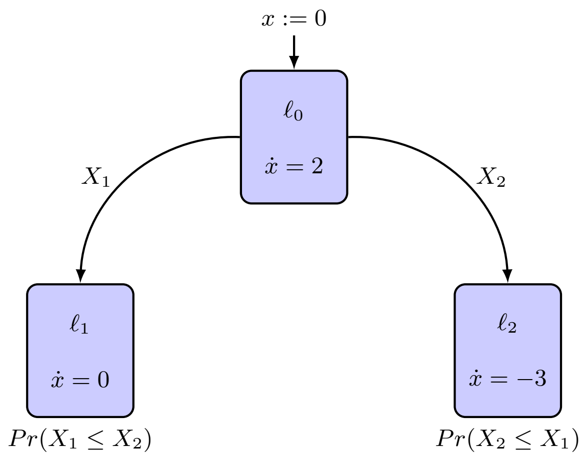

Figure 4: Case A. Simple stochastic hybrid automaton with two random variables.

location ℓ_2 is reached, which can only happen, if $X_2 < X_2$. Due to the time bound of the property, location ℓ_0 has to be left within 5.8 time units. The probability of ϕ hence equals $Pr(X_2 < X_1) \cdot Pr(X_2 \leq 5.8)$.

4.3.2 Case B

In this case, the initial stochastic choice between location ℓ_1 and ℓ_2 is the same as in case A. However, in location ℓ_2 the variable x does not evolve and the goal valuation for x cannot be reached here. Instead, there is an invariant connected to an additional continuous variable y, which evolves with rate 1. Due to this invariant and the guards of both outgoing edges, ℓ_2 has to be left after exactly 2 time units. As two transitions are enabled at $t = 2$, there is a conflict between the transitions to location ℓ_3 and ℓ_4. An illustration of case B is provided in Figure 5.

We want to check whether the continuous variable x reaches the valuation of -1 within a time bound of 10 (as before) and with a time bound of 12:

$$\phi' = F^{\leq 12}(x \leq -1).$$

When maximizing, the probability to fulfil property ϕ' should be equal to the probability of property ϕ for case A, since there is an additional delay of two time units in location ℓ_2. If the discrete non-determinism is resolved probabilistically and both transitions have the same probability, the probability is expected to be half the one computed for case A. Accordingly, for property ϕ, the probability is expected to be smaller, since there is less time to reach the goal valuation due to the time delay in ℓ_2.

4.3.3 Case C

The automaton for case C is very similar to the one in case B (see Figure 5). However, we assume the evolution of y in location ℓ_2 to be stochastically disturbed by a $\mathcal{N}(0, 2)$ distributed random variable n_1. Thus $\dot{y} = 1 + n_1$.

Again, we propose to compute results for formulas ϕ and ϕ'.

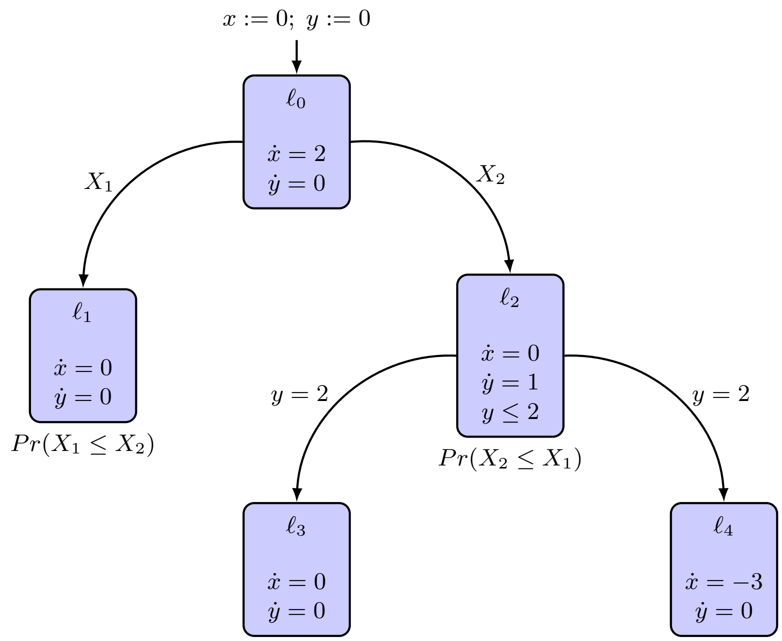

Figure 5: Case B. Simple stochastic hybrid automaton with two random variables as well as a pair of an invariant and corresponding guards.

4.3.4 Case D

This example is an extension of case A, hence, the time spent in location ℓ_0 depends on the two random variables X_1 and X_2. However, there is an invariant in $x \leq 6$ in the initial location. This leads to the case that none of the transitions can be taken if both delays corresponding to the random variables are larger than 3, which results in a time lock. The illustration is given in Figure 6.

To compare how different approaches and tools handle this situation, we want to compute the probability of the established formula ϕ:

$$\phi = F^{\leq 10}(x \leq -1).$$

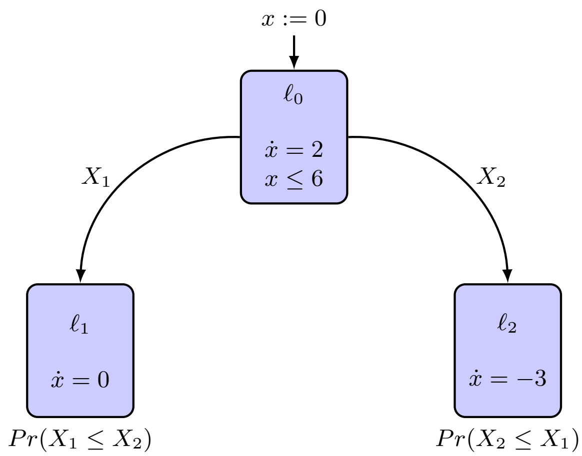

Figure 6: Case D. Simple stochastic hybrid automaton with an invariant resulting in a time lock.

Table 3: 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).

Tool	Benchmarks										
	AS	BA	HT	WS	VP	IC	AV	PR	GB	ME	PD
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
Figaro			2021								
PyCATSHOO			2021								

Table 4: 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 : Brownian motion (BM) or independently and identically distributed (iid)

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

5 Friendly Competition – Setup and Outcomes

5.1 Building automation benchmark results

Table 5 compares the performance of the tools based on their run time and the highest stochastic reach probability starting from any state in the initial safe set for the building automation benchmark. The benchmark defines a stochastic viability problem for a four-dimensional and seven-dimensional Gaussian-perturbed LTI system model (see Section 3.1).

SySCoRe has been applied to the 7-dimensional version of this benchmark. SySCoRe uses both model reduction and space discretization to compute a bound for the reach probability. The dimension of the system is reduced from 7 to 2. The lowerbound on the Maximum reach probability is equal to 0.9035 with the total run time 83.35 seconds. These values are obtained by setting $(\epsilon, \delta) = (0.35, 0.0161)$.

Table 5

Property	StocHy	AMYTISS	SySCoRe
Case 1, 4-dimensional system			
Run time on common CPU(sec)	7.17	0.92	*not tested*
Maximum reach probability	$\geq 0.99 \pm 0.05$	≈ 0.99	*not tested*
Case 2, 7-dimensional system			
Run time on common CPU (sec)	335.876	12.5	83.35
Maximum reach probability	$\geq 0.8 \pm 0.23$	≈ 0.8	≥ 0.9035

5.2 Heated Tank benchmark results

Both in ARCH2018, ARCH2019 and ARCH2020 the focus has been on the estimation of the dryout probability for Heated Tank version 4.0 [4, 2]. Within ARCH2021 the objectives has been to evaluate Heated Tank version 4.III. The extension of the formal model specification of HT version 4.0 to HT version 4.III consists of the following three extensions: i) Change in differential equation for the temperature $x_{T,t}$; ii) Temperature dependent failure rates of Valve and Pumps; and iii) Change in model parameter values. None of these extensions impact the graphical Petri Net model of version 4.0 [2]. The details of these exyensions have been specified in subsection 5.4 of [1]. In 2021 and 2022, Heated Tank version 4.III has been evaluated by FIGARO, PyCATSHOO, SDCPN&MC, SDCPN&IPS, SDCPN&IS, and HYPEG. The results obtained for Dry-out probability are given in Table 6Table 6 presents the results obtained in 2021.It should be noted that in case of reaching Boiling prior to reaching the Dryout level, the simulations are continued; this is indicated as P-Dryout non-stop.

The FIGARO tools used for HT 4.III are: FigaroIDE to build a small knowledge base, KB3 to input the system graphically, the Figaro0 language for a self-contained model, and YAMS for running Monte Carlo simulations. For the numerical solution of the differential equations a forward Euler method is used with a fixed time step. Conducting 1 million runs asked 3 h53 min on an Intel Core i5-6200U, 2.3Ghz Processor with a time step of 0.5h, and 18h40mn with a time step of 0.1h. The reduction of the time step increases precision, this is why Table 6 only contains the result obtained with the time step of 0.1h. But it is interesting to note that taking a larger time step leads to an overestimation of the probability (9.4×10^{-5} with time step 0.5h).

The PyCATSHOO model for the HT 4.III is based on four concrete python classes: Tank, Pump, Valve and ThermalSource. Each one of these classes is modelled by automata, by a set of state variables, and by equations that govern these variables. These classes provide message boxes where incoming and outgoing channels are used as a means of communication between the interconnected objects in the system. As the PyCASTHOO acceleration mechanism [25] is still under development, we first used straightforward Monte Carlo simulations. This gave us a comparison benchmark to confirm the result of our importance sampling (IS) algorithm. By using an EDF high-performance computer, it was feasible to conduct 1 million straightforward MC runs in 19 seconds. On a laptop with i7-8750H CPU @ 2.2 GHz, this required about 50mn.

The SDCPN model for HT 4.III has been realized by extending the SDCPN model for HT 4.0 that has been used in [2]. For the numerical evaluation of the differential equations in between stopping times, a forward Euler method is used with a time step of 0.1 hour (or less). The number of MC runs is 10 million. The number of IPS runs is 100, and the number of particles per IPS run is 100 thousand. The MC and IPS runs have been conducted on an ASUS RS700A-E9-RS4 with an AMD Epyc 7551 processor having 32 cores and 64 threads and

Table 6: P-Dryout for Heated Tank version 4.III: estimated by FIGARO, PyCATSHOO, SDCPN & MC and SDCPN & IPS (source [1])

Method / Measure	FIGARO	PyCATSHOO		SDCPN&MC	SDCPN&IPS
Variance reduction	No	No	IS	No	IPS
Estimated P-Dryout non-stop	5.6×10^{-5}	2.40×10^{-5}	2.86×10^{-5}	1.98×10^{-5}	1.99×10^{-5}
Confidence interval	$\pm 1.46 \times 10^{-5}$ (95%)	$\pm 0.96 \times 10^{-5}$ (95%)		$\pm 0.28 \times 10^{-5}$ (95%)	$\pm 0.041 \times 10^{-5}$ (95%)
Simulation effort	10^6 runs	10^6 runs	15000 runs	10^7 runs	100 x IPS a 100,000 part.

Table 7: Comparison of different methods in simulating spontaneous jumps in IPS based estimation of P-Dryout non-stop for the Heated Tank version 4.III.

Method in simulating spontaneous jumps during IPS based estimation	SDCPN&IPS Method 0	SDCPN&IPS Method 1	SDCPN&IPS Method 2
Variance reduction	IPS	IPS	IPS
Estimated P-Dryout non-stop	2.07×10^{-5}	1.99×10^{-5}	1.97×10^{-5}
Confidence interval (95%)	$\pm 0.149 \times 10^{-5}$	$\pm 0.041 \times 10^{-5}$	$\pm 0.039 \times 10^{-5}$
Simulation effort	100 x IPS a 100,000 part.	100 x IPS a 100,000 part.	100 x IPS a 100,000 part.
Computer time	1.42 hours	2.52 hours	1.32 hours

256 GB of RAM. The 10 million MC runs asked 1.37 hour; the 100 IPS runs asked 2.52 hour. Comparison of the estimation results of SDCPN&MC versus SDCPN&IPS shows that their estimated P-Dryout probabilities are almost the same, though the 95% uncertainty interval of IPS is about a factor 7 smaller than it is for MC.

The estimated P-Dryout probabilities by PyCATSHOO, SDCPN&MC and SDCPN&IPS fall outside the 95% confidence interval of FIGARO. The likely explanation is that the former three used a discrete event simulation method, i.e. to apply a numerical integration method in between two successive stopping times of the process to be simulated, whereas FIGARO used fixed time steps of 0.1 hour, which means that a stopping time of the process to be simulated may be somewhere halfway an integration time step instead of being at the begin or end. This difference, and also the difference between the two results obtained with the same FIGARO model with different time steps shows that the apparently simple Heated tank benchmark is sensitive to numerical approximation.

Table 7 presents new benchmark results obtained in 2022 using SDCPN&IPS. These novel results show that the specific Monte Carlo simulation method that is used to generate spontaneous jumps may have significant effect on the IPS results for the Heated Tank 4.III benchmark. In simulating a spontaneous jump in a PDMP or GSHS, the common practice (Method 0) is to draw a random delay sample from the probability density function of the next spontaneous jump. This random delay sample forms a realization of the remaining time until the next spontaneous jump.

During the subsequent simulation, this remaining time sample counts down in time, and upon reaching zero remaining time, the spontaneous jump is realized in the Monte Carlo simulation. Both [8] and [24] have shown that this common approach (Method 0) in simulating spontaneous jumps does not work well in combination with IPS. Two general methods in mitigating this problem are:

- Method 1: To draw a random delay sample, and to start counting the time passed since this drawing. If the time counter equals the value of the random sample, then the spontaneous jump is implemented in the MC simulation [8].

- Method 2: To apply the common method, i.e. draw random delay samples and discount time until level zero has been reached, though draw new random delay samples at the beginning of each new IPS cycle [58].

Method 1 has been used to get the results in the SDCPN&IPS column in Table 6. Table 7 shows what happens when instead of Method 1, the common Method 0 and the recent Method 2 are used. The results in Table 7 show that the common Method 0 suffers from a significant factor less good IPS variance reduction than methods 1 and 2 do. Regarding variance reduction, Method 1 and Method 2 perform similarly well. However, an advantage of Method 2 over Method 1 is that its computational load is significantly lower (1.32 hours simulation time for Method 2 versus 2.52 hours for Method 1).

5.3 Van der Pol Oscillator benchmark results

Tool SySCoRe has been applied to Problem 5. To synthesize a controller for a nonlinear system, SySCoRe performs a piecewise-affine approximation and quantifies the additional error. This method is discussed in detail in [76]. The total computation time on CodeOcean is 3690 seconds.

5.4 Package Delivery

Tool SySCoRe has been applied to the package delivery benchmark as described in Sec. 4.1. The tool generates a satisfying controller in 12.93 seconds. Fig. 7 displays the obtained satisfaction probability conditioned on the initial state.

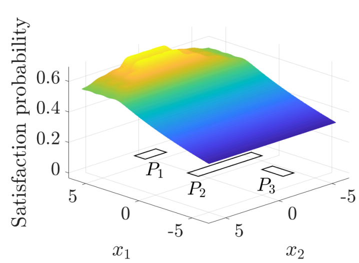

Figure 7: Lower bound on the satisfaction probability as a function of the initial state for the package delivery study.

Figure 8: Results SySCoRe

Run time on common CPU(sec)	12.929
Maximum reach probability	0.6632

5.5 Minimal examples results

For the computation of results for the minimal examples, random variables X_1 and X_2 can follow different distributions. We compared results for two sets of continuous distributions:

(1) $X_1 \sim exp(0.1), X_2 \sim exp(0.08)$,

(2) $X_1 \sim exp(0.1), X_2 \sim \mathcal{N}(5, 2)$.

In (1), both random variables follow an exponential distribution; the time delay in location ℓ_0 can hence be modeled by one random variable $exp(0.1 + 0.08)$. This is not possible in (2), as X_1 follows an exponential distribution and X_2 a folded normal distribution.

Results for (1) are presented in Table 8 and for (2) in Table 9. Results for case C have not been included, since none of the participating tools are 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 seperately 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 the nondeterminism is resolved probabilistically.

Computation HYPEG was used to estimate the probabilities of the fully stochastic models A and D. In the default setting, HYPEG resolves nondeterminism (contained in case B) uniformly which is included in the first row (indicated by "**SMC**"). By applying Q-learning (indicated by "**Q-learn**"), nonprophetic memoryless schedulers are trained to maximize or minimize the probability of ϕ and ϕ'. 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 indicated in the first row 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.

modes has been run in four modes concerning its handling of nondeterminism: Cases A and D are fully stochastic, and thus modes needs no further configuration to handle these models. It would by default abort upon detecting nondeterminism; as expected, it does not do so in these cases, correctly producing estimates of the probabilities. For case B, we first resolve the nondeterministic choice uniformly at random (i.e. as a 50/50 random choice), resulting in an estimate of some probability between the maximum and minimum. In the tables, we include these two configurations in the same row, marked with the "probabilistic" resolution of nondeterminism. We then apply LSS to case B, sampling 10000 schedulers from the space of deterministic history-dependent schedulers (rows marked with "hist" in the tables), and from the space of all deterministic memoryless schedulers (rows marked with "ml"). While the latter is in principle less powerful [28], reducing the sampling space may also increase the likelihood of finding a good scheduler among the remaining ones, leading to a better probability overall. We note that modes checks both properties of case B in one go; thus the runtime R indicated for each is actually the total runtime for both properties (marked as "$< R$s" for emphasis). The statistical evaluation of modes was configured to perform as many runs as necessary to achieve an absolute error of at most ± 0.01 in 95 % of the tool invocations. To achieve repeatable results, however, we fixed the seeds for modes' pseudo-random number generators.

For prohver, we applied the new automated interval refinement method to deal with the continuous random variables. Where both are exponentially distributed, we also use two variants of the model: One where the variables have been merged into one with the sum of the rates (indicated by "M" in Table 8), and one where they remain separate (indicated by "S"). We used the following initial intervals:

- $[0, 10), [10, \infty)$ for the exponential sampling in the merged exponential variant,
- $[0, 5), [5, \infty)$ for each of the two variables in the split variant with two exponentials, and
- $[0, 5), [5, \infty)$ for the exponential and $[0, 5), [5, 10), [10, \infty)$ for the folded normal sampling in the variant with the folded normal distribution.

For each case, we instructed prohver to refine to at most 100 intervals, using three different splitting factors: we split the best interval in each step into 2, 4, or 8 new intervals. Like modes, prohver checks both properties of case B in one tool invocation, so we mark the runtimes in the same way.

RealySt is specifically designed to resolve nondeterminism prophetically in rectangular automata with random clocks. The tool can also compute reachability probabilities in fully stochastic models. The minimal examples are all singular automata with random clocks, where case B exhibits discrete nondeterminism via a choice between two transition jumps. Reachable state sets are computed exactly in convex polytope representation. Then, the dedicated integration method via Monte Carlo Vegas is called for a predefined number of integration samples and a fixed integration bound. A larger number of integration samples reduces the statistical error (indicated by "stat:"). The integration bound limits the domain over which is integrated and all probability mass after the integration bound will be cut off. The maximal cut off probability mass is stated as a max error (indicated by "max:"). We used 100000 integration samples and an integration bound of 100. In contrast to prohver, we do not need to further discretise the domain of the random variables. However, the performance of RealySt highly depends on the dimension of the underlying state-space, which is given by the sum of the continuous and the random variables.

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. modes and prohver ran on an Intel Core i7-1185G7 system with 32 GB of RAM inside the Windows Subsystem for Linux on 64-bit Windows 10. RealySt was executed on a machine with a 2.50GHz Intel i5-7200U CPU and 16 GB of RAM.

Discussion As for the Modest Toolset, we see in the results that modes is effective and efficient in the fully stochastic cases. For the nondeterministic model, LSS works well—but we caution that this is on an extremely simple model, where it is relatively likely to randomly sample a good scheduler. Where modes with LSS underapproximates the maximum probability, prohver overapproximates it. By comparing with the other tools, we find that it manages to find good approximations in most cases, albeit at a somewhat higher computational effort. The new interval refinement technique, while conceptually simple, works very well—though again, these are extremely simple models. In setting up the experiments, we in fact noticed that the procedure is very sensitive to the choice of initial intervals, terminating too early for some choices and taking a very long time to obtain any improvements for other choices.

Comparing statistical model checking (SMC, prob) with HYPEG and modes, it can be seen that results align well and that HYPEG is faster. In case B, HYPEG maximized and minimized the probabilities. When minimizing, overapproximations are computed, in case of maximizing, similar to modes with LSS, underapproximations are obtained. The computed confidence intervals in Table 8 of HYPEG for the maximum case overlap the confidence intervals obtained by modes with LSS and memoryless schedulers, regardless of the M or S model variant. The same behaviour can be seen in Table 9, where again the results for the maximizing memoryless schedulers match.

As expected, prohver overapproximates the results computed by RealySt. For all cases shown in Table 8, the difference between the RealySt result and the best prohver result (**ref-8**) lies in the order of 10^{-3}, whereas the RealySt error is in the order of 10^{-4}. The results shown in Table 9 are computed for the combination of a folded normal and exponential distribution. Here, the difference between the RealySt results and the best prohver result (**ref-8**) is in the order of 10^{-1}, while the RealySt error is in 10^{-5}. In both scenarios, RealySt is much faster. All results have been computed in less than a second by RealySt, which is significantly faster than prohver.

When maximizing the probability that ϕ' holds in case B, both analytical tools compute the same values as for the probability that ϕ holds in case A. This was to be expected, as the time bound in ϕ' is two time units larger and hence compensates the additional delay in case B.

Comparing the results of all tools in the fully stochastic cases A and D, the results of RealySt lie in the respective confidence intervals of modes and HYPEG and prohver provides overapproximations which lie in Table 8 in some configurations also within the confidence intervals which is not the case in Table 9. In case B, similar results can be obtained, even though RealySt used prophetic schedulers and the other tools nonprophetic schedulers. This lies in the simplicity of the models, as the additional information on the random variables does not have an impact on the optimal decisions.

Table 8: Results for minimal examples with random variables X_1 and X_2 following distributions $exp(0.1)$ and $exp(0.08)$. Results contain the computed probability, error(s) or confidence interval if available, and computation times.

Tools		Case / Formula			
Tool	Method	A / ϕ	B / ϕ	B / ϕ'	D / ϕ
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	
modes	**SMC**, **prob**, M	≈ 0.284034 ± 0.01 @ 95 % 0.860s	≈ 0.119730 ± 0.01 @ 95 % < 0.91s	≈ 0.136825 ± 0.01 @ 95 % < 0.91s	≈ 0.188073 ± 0.01 @ 95 % 0.91s
	LSS/hist, **max**, M		$\approx 0.225664 \leq$ max ± 0.01 @ 95 % < 1.70s	$\approx 0.241987 \leq$ max ± 0.01 @ 95 % < 1.70s	
	LSS/ml, **max**, M		$\approx 0.256704 \leq$ max ± 0.01 @ 95 % < 1.88s	$\approx 0.291650 \leq$ max ± 0.01 @ 95 % < 1.88s	
	SMC, **prob**, S	≈ 0.289604 ± 0.01 @ 95 % 0.91s	≈ 0.122316 ± 0.01 @ 95 % < 0.91s	≈ 0.141631 ± 0.01 @ 95 % < 0.91s	≈ 0.189252 ± 0.01 @ 95 % 0.90s
	LSS/hist, **max**, S		$\approx 0.222256 \leq$ max ± 0.01 @ 95 % < 1.60s	$\approx 0.267302 \leq$ max ± 0.01 @ 95 % < 1.60s	
	LSS/ml, **max**, S		$\approx 0.254877 \leq$ max ± 0.01 @ 95 % < 1.81s	$\approx 0.285518 \leq$ max ± 0.01 @ 95 % < 1.81s	
prohver	**ref-2**, **max**, M	$0.317109 \geq$ max 0.99s	$0.317109 \geq$ max < 1.35s	$0.317109 \geq$ max < 1.35s	$0.206551 \geq$ max 2.33s
	ref-4, **max**, M	$0.288125 \geq$ max 2.97s	$0.256255 \geq$ max < 6.18s	$0.288125 \geq$ max < 6.18s	$0.206551 \geq$ max 1.41s
	ref-8, **max**, M	$0.295575 \geq$ max 2.31s	$0.270398 \geq$ max < 4.71s	$0.295575 \geq$ max < 4.71s	$0.187220 \geq$ max 2.33s
	ref-2, **max**, S	$0.297317 \geq$ max 28.1s	$0.265495 \geq$ max < 61.6s	$0.297317 \geq$ max < 61.6s	$0.191175 \geq$ max 29.3s
	ref-4, **max**, S	$0.292234 \geq$ max 2603s	$0.253392 \geq$ max < 5295s	$0.292234 \geq$ max < 5295s	$0.187469 \geq$ max 2341s
	ref-8, **max**, S	$0.295300 \geq$ max 1152s	$0.254787 \geq$ max < 2297s	$0.295300 \geq$ max < 2297s	$0.189655 \geq$ max 1020s
RealySt	**max**	0.288236 stat: $4.651 \cdot 10^{-4}$ max: $3.808 \cdot 10^{-4}$ 0.307s	0.250016 stat: $3.239 \cdot 10^{-4}$ max: $3.808 \cdot 10^{-4}$ 0.680226s	0.288236 stat: $4.651 \cdot 10^{-4}$ max: $3.808 \cdot 10^{-4}$ 0.362251s	0.185280 stat: $2.061 \cdot 10^{-4}$ max: $3.808 \cdot 10^{-4}$ 0.351297s

Table 9: Results for minimal examples with random variables X_1 and X_2 following distributions $exp(0.1)$ and $\mathcal{N}(5, 2)$. Results contain the computed probability, error(s) or confidence interval if available, and computation times.

Tools		Case / Formula			
Tool	Method	A / ϕ	B / ϕ	B / ϕ'	D / ϕ
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	
modes	**SMC**, **prob**	≈ 0.445460 ± 0.01 @ 95 % 0.90s	≈ 0.155145 ± 0.01 @ 95 % < 0.88s	≈ 0.222971 ± 0.01 @ 95 % < 0.88s	≈ 0.130161 ± 0.01 @ 95 % 0.91s
	LSS/hist, **max**		$\approx 0.298658 \leq$ max ± 0.01 @ 95 % < 1.80s	$\approx 0.402019 \leq$ max ± 0.01 @ 95 % < 1.80s	
	LSS/ml, **max**		$\approx 0.299426 \leq$ max ± 0.01 @ 95 % < 1.80s	$\approx 0.447541 \leq$ max ± 0.01 @ 95 % < 1.80s	
prohver	**ref-2**, **max**	$0.734016 \geq$ max 1.01s	$0.500000 \geq$ max < 1.68s	$0.734016 \geq$ max < 1.68s	$0.500000 \geq$ max 0.54s
	ref-4, **max**	$0.655567 \geq$ max 9.12s	$0.500001 \geq$ max < 20.8s	$0.655567 \geq$ max < 20.8s	$0.158979 \geq$ max 9.65s
	ref-8, **max**	$0.655566 \geq$ max 22.38s	$0.500000 \geq$ max < 49.4s	$0.655566 \geq$ max < 49.4s	$0.164365 \geq$ max 21.2s
RealySt	**max**	0.448211 stat: $8.292 \cdot 10^{-5}$ max: $2.061 \cdot 10^{-9}$ 0.18043s	0.308558 stat: $5.221 \cdot 10^{-5}$ max: $2.061 \cdot 10^{-9}$ 0.115976s	0.448211 stat: $8.292 \cdot 10^{-5}$ max: $2.061 \cdot 10^{-9}$ 0.108529s	0.130280 stat: $1.766 \cdot 10^{-5}$ max: $2.061 \cdot 10^{-9}$ 0.792725s

6 Conclusions

The evaluation of benchmarks this year featured six tools, among these a novel tool (RealySt). Apart from regular operation, i.e., evaluating benchmarks, we welcome the special initiative this year which targeted development of a set of minimal benchmarks to allow comparison of tools and their implemented approaches. The result of this initiative feature four new minimal benchmarks. Additionally, another novel benchmark (*package delivery system*) was added to our collection.

In the following, we give the tool authors space to describe planned future developments for their tools which we are looking forward to see in the next years as part of this subgroup.

6.1 Further tool development

AMYTISS, StocHy, and FAUST^2 did not participate in this year competition due to lack of supporting hybrid models. They will join the competition next year with further development on verification and synthesis of stochastic *hybrid* systems potentially for infinite horizon properties [73, 74].

The tool Mascot-SDS [62, 61, 60, 5, 6] is currently only handling synthesis of formally verified controllers for almost sure satisfaction (i.e. satisfaction with probability 1) of infinite-horizon specifications. Further work include applying Mascot-SDS to the Package Delivery benchmark, and implementing the quantitative aspect of the synthesis problem (i.e. computing the optimal probability of satisfying the specification for any initial state).

In the Modest Toolset, we plan to integrate the reinforcement learning-based approach that was implemented as a prototype for [63] as a fully developed method in modes. For prohver, the simple interval refinement approach added specifically for this competition showed some promise, but requires a lot of refinement to become a robust method that users can rely on to automatically deliver good approximations.

We plan to integrate in HYPEG the possibility to support failure rates depending on continuous variables to be able to participate in the Heated Tank version 4.III benchmark.

6.1.1 Further development of SySCoRe

To increase the range of benchmarks that can be handled by SySCoRe, we want to extend the tool to other distributions, namely distributions with a bounded support (e.g., uniform distributions). The current implementation can only handle nonlinear systems with an affine input. We plan to extend the tool and the underlying techniques to fully nonlinear systems. We also plan to improve the computation time by implementing parallel computations in future versions of the tool.

6.1.2 Further development of RealySt

RealySt is currently being developed within the DFG project 471367371 as a cooperation between RWTH Aachen University and the University of Münster. In the coming years, we want to add the possibility to maximize and minimize reachability probabilities for rectangular hybrid automata with random clocks. Furthermore, we will investigate how to make the numerical integration faster and more accurate and which other state-space representations are suitable. Finally, we plan to include the possibility to parse models specified in the JANI specification language [18].

References

[1] Alessandro Abate, Henk Blom, Marc Bouissou, Nathalie Cauchi, Hassane Chraibi, Joanna Delicaris, Sofie Haesaert, Arnd Hartmanns, Mahmoud Khaled, Abolfazl Lavaei, Hao Ma, Kaushik Mallik, Mathis Niehage, Anne Remke, Stefan Schupp, Fedor Shmarov, Sadegh Soudjani, Adam Thorpe, Vlad Turcuman, and Paolo Zuliani. Arch-comp21 category report: Stochastic models. In Goran Frehse and Matthias Althoff, editors, *8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)*, volume 80 of *EPiC Series in Computing*, pages 55–89. EasyChair, 2021.

[2] Alessandro Abate, Henk Blom, Nathalie Cauchi, Kurt Degiorgio, Martin Fraenzle, Ernst Moritz Hahn, Sofie Haesaert, Hao Ma, Meeko Oishi, Carina Pilch, Anne Remke, Mahmoud Salamati, Sadegh Soudjani, Birgit van Huijgevoort, and Abraham Vinod. ARCH-COMP19 category report: Stochastic modelling. In *ARCH19. 6th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 61 of *EPiC Series in Computing*, pages 62–102. EasyChair, 2019.

[3] Alessandro Abate, Henk Blom, Nathalie Cauchi, Joanna Delicaris, Arnd Hartmanns, Mahmoud Khaled, Abolfazl Lavaei, Carina Pilch, Anne Remke, Stefan Schupp, Fedor Shmarov, Sadegh Soudjani, Abraham Vinod, Ben Wooding, Majid Zamani, and Paolo Zuliani. Arch-comp20 category report: Stochastic models. In Goran Frehse and Matthias Althoff, editors, *ARCH20. 7th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH20)*, volume 74 of *EPiC Series in Computing*, pages 76–106. EasyChair, 2020.

[4] Alessandro Abate, Henk Blom, Nathalie Cauchi, Sofie Haesaert, Arnd Hartmanns, Kendra Lesser, Meeko Oishi, Vignesh Sivaramakrishnan, Sadegh Soudjani, Cristian-Ioan Vasile, and Abraham P. Vinod. ARCH-COMP18 category report: Stochastic modelling. In *ARCH18. 5th International Workshop on Applied Verification of Continuous and Hybrid Systems*, volume 54 of *EPiC Series in Computing*, pages 71–103. EasyChair, 2018.

[5] Tamajit Banerjee, Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. A direct symbolic algorithm for solving stochastic rabin games. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 81–98. Springer, 2022.

[6] Tamajit Banerjee, Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Fast symbolic algorithms for omega-regular games under strong transition fairness. *arXiv preprint arXiv:2202.07480*, 2022.

[7] H.A.P. Blom, J. Krystul, G.J. Bakker, M.B. Klompstra, and B. Klein Obbink. Free flight collision risk estimation by sequential mc simulation. In C.G. Cassandras and J. Lygeros, editors, *Stochastic hybrid systems*, chapter 10, pages 249–281. Taylor & Francis/CRC Press, 2007.

[8] H.A.P. Blom, H. Ma, and G.J. Bakker. Interacting particle system-based estimation of reach probability for a generalized stochastic hybrid system. In *Proc. Conference Analysis and Design of Hybrid Systems (ADHS 2018)*, volume 51, pages 79–84. IFAC Papers Online 51-16, Oxford, UK, 2018.

[9] J.-L. Bon and J. Collet. An algorithm in order to implement reliability exponential approximations. *Reliability Engineering & System Safety*, 43(3):263–268, 1994.

[10] M. Bouissou. A simple yet efficient acceleration technique for Monte Carlo simulation. In *Proceedings of the 22nd European Safety and Reliability Conference (ESREL'13)*, page 27–36, 2013.

[11] M. Bouissou and J.C. Houdebine. Inconsistency detection in KB3 models. *ESREL 2002*, 2002.

[12] M. Bouissou, J.C. Houdebine, and Humbert S. Reference manual of the Figaro probabilistic modelling language. 2019.

[13] M. Bouissou and S. Khan. Bridging the Dependability and Model Checking worlds. In *Proc. of Lambda-mu 23*, October 2022.

[14] M. Bouissou and Y. Lefebvre. A path-based algorithm to evaluate asymptotic unavailability for large markov models. In *Proceedings of RAMS'2002*, pages 32–39, 2002.

[15] Carlos E. Budde, Pedro R. D'Argenio, and Arnd Hartmanns. Automated compositional importance

splitting. *Sci. Comput. Program.*, 174:90–108, 2019.

[16] Carlos E. Budde, Pedro R. D'Argenio, Arnd Hartmanns, and Sean Sedwards. An efficient statistical model checker for nondeterminism and rare events. *Int. J. Softw. Tools Technol. Transf.*, 2020. to appear.

[17] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: Quantitative model and tool interaction. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 151–168, 2017.

[18] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: quantitative model and tool interaction. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 151–168, 2017.

[19] M. L. Bujorianu and J. Lygeros. Toward a general theory of stochastic hybrid systems. In H.A.P. Blom and J. Lygeros, editors, *Stochastic hybrid systems*, pages 3–30. Springer, Berlin, 2006.

[20] Nathalie Cauchi and Alessandro Abate. Benchmarks for cyber-physical systems: A modular model library for buildings automation. In *IFAC Conference on Analysis and Design of Hybrid Systems*, 2018.

[21] Nathalie Cauchi and Alessandro Abate. StocHy: automated verification and synthesis of stochastic processes. In *25th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, 2019.

[22] Nathalie Cauchi, Luca Laurenti, Morteza Lahijanian, Alessandro Abate, Marta Kwiatkowska, and Luca Cardelli. Efficiency through uncertainty: Scalable formal synthesis for stochastic hybrid systems. In *22nd ACM International Conference on Hybrid Systems: Computation and Control (HSCC)*, 2019. arXiv: 1901.01576.

[23] F. Cérou, P. Del Moral, F. Legland, and P. Lezaud. Genetic genealogical models in rare event analysis. *Latin American J. of Probability and Mathematical Statistics*, 1:181–203, 2006.

[24] H. Chraibi, A. Dutfoy, T. Galtier, and J. Garnier. Application of interacting particle system method to piecewise deterministic markov processes used in reliability, preprint submitted to Chaos, 23rd may 2019. arxiv:1905.09044v1.

[25] H. Chraibi, A. Dutfoy, T. Galtier, and J. Garnier. On the optimal importance process for piecewise deterministic markov process. *ESAIM: Probability and Statistics*, 23:893–921, 2019.

[26] H. Chraibi, J.C. Houbedine, and A. Sibler. PyCATSHOO: Toward a new platform dedicated to dynamic reliability assessments of hybrid systems. In *13th International Conference on Probabilistic Safety Assessment and Management (PSAM 13)*, Seoul, Korea, 2016.

[27] D. Codetta-Raiteri. Modelling and simulating a benchmark on dynamic reliability as a stochastic activity network. In *23rd European Modeling and Simulation Symposium (EMSS)*, pages 545–554, 2011.

[28] Pedro R. D'Argenio, Marcus Gerhold, Arnd Hartmanns, and Sean Sedwards. A hierarchy of scheduler classes for stochastic automata. In Christel Baier and Ugo Dal Lago, editors, *Foundations of Software Science and Computation Structures - 21st International Conference, FOSSACS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings*, volume 10803 of *Lecture Notes in Computer Science*, pages 384–402. Springer, 2018.

[29] M.H.A. Davis. *Markov models and optimization.* Chapman and Hall, London, 1993.

[30] M.H.C. Everdij and H.A.P. Blom. Piecewise deterministic Markov processes represented by dynamically coloured Petri nets. *Stochastics*, 77:1–29, 2005.

[31] M.H.C. Everdij and H.A.P. Blom. Bisimulation relations between automata, stochastic differential equations and Petri nets. In M. Bujorianu and M. Fisher, editors, *Workshop on Formal Methods for Aerospace (FMA), Electronic Proceedings in Theoretical Computer Science, EPTCS 20*, page 1–15, 2010.

[32] M.H.C. Everdij and H.A.P. Blom. Hybrid state Petri nets which have the analysis power of stochastic hybrid systems and the formal verification power of automata. In P. Pawlewski, editor, *Petri Nets*, chapter 12, pages 227–252. I-Tech Education and Publishing, Vienna, 2010.

[33] M.H.C. Everdij, M.B. Klompstra, H.A.P. Blom, and B. Klein Obbink. Compositional specification of a multi-agent system by stochastically and dynamically coloured Petri nets. In J. Lygeros H.A.P. Blom, editor, *Stochastic Hybrid Systems: Theory and safety critical applications*, pages 325–350. Springer, 2006.

[34] Martin Fränzle, Ernst Moritz Hahn, Holger Hermanns, Nicolás Wolovick, and Lijun Zhang. Measurability and safety verification for stochastic hybrid systems. In Marco Caccamo, Emilio Frazzoli, and Radu Grosu, editors, *Proceedings of the 14th ACM International Conference on Hybrid Systems: Computation and Control, HSCC 2011, Chicago, IL, USA, April 12-14, 2011*, pages 43–52. ACM, 2011.

[35] Goran Frehse. PHAVer: algorithmic verification of hybrid systems past HyTech. *Int. J. Softw. Tools Technol. Transf.*, 10(3):263–279, 2008.

[36] Brian Gough. *Gnu Scientific Library Reference Manual*. Network Theory Ltd., 2009.

[37] Marco Gribaudo and Anne Remke. Hybrid Petri nets with general one-shot transitions. *Performance Evaluation*, 105:22–50, 2016.

[38] Sofie Haesaert, Petter Nilsson, and Sadegh Soudjani. Formal multi-objective synthesis of continuous-state MDPs. *IEEE Control Systems Letters*, 5(5):1765–1770, 2020.

[39] Sofie Haesaert and Sadegh Soudjani. Robust dynamic programming for temporal logic control of stochastic systems. *IEEE Transactions on Automatic Control*, 66(6):2496–2511, 2020.

[40] Sofie Haesaert, Sadegh Soudjani, and Alessandro Abate. Temporal logic control of general Markov decision processes by approximate policy refinement. *IFAC-PapersOnLine*, 51(16):73–78, 2018.

[41] Sofie Haesaert, Sadegh Esmaeil Zadeh Soudjani, and Alessandro Abate. Verification of general Markov decision processes by approximate similarity relations and policy refinement. *SIAM Journal on Control and Optimization*, 55(4):2333–2367, 2017.

[42] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design*, 43(2):191–232, 2013.

[43] Arnd Hartmanns and Holger Hermanns. The Modest Toolset: An integrated environment for quantitative modelling and verification. In *20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 8413 of *Lecture Notes in Computer Science*, pages 593–598. Springer, 2014.

[44] Jannik Hüls, Carina Pilch, Patricia Schinke, Henner Niehaus, Joanna Delicaris, and Anne Remke. State-space Construction of Hybrid Petri Nets with Multiple Stochastic Firings. *ACM Transactions on Modeling and Computer Simulation*, 31(3):1–37, 2021.

[45] 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.

[46] Jannik Hüls, Carina Pilch, Patricia Schinke, Joanna Delicaris, and Anne Remke. State-Space Construction of Hybrid Petri Nets with Multiple Stochastic Firings. In *16th International Conference on Quantitative Evaluation of Systems, QEST 2019*, volume 11785 of *LNCS*, pages 182–199. Springer, 2019.

[47] 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*, volume 11535 of *LNCS*, pages

148–166, Cham, 2019. Springer.

[48] Jannik Hüls, Stefan Schupp, Anne Remke, and Erika Ábrahám. Analyzing Hybrid Petri nets with multiple stochastic firings using HyPro. In *11th EAI International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2017*, pages 178–185. ACM, 2018.

[49] S. Khan, J.-P. Katoen, M. Volk, and M. Bouissou. Scalable Reliability Analysis by Lazy Verification. In *Proc of NASA Formal Methods: 13th International Symposium, NFM 2021*. Springer, 2021.

[50] S. Khan, M. Volk, J.P. Katoen, A. Braibant, and M. Bouissou. Model checking the multi-formalism language Figaro. In *Proceedings of DSN 2021*, 2021.

[51] Marta Z. Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theor. Comput. Sci.*, 282(1):101–150, 2002.

[52] A. Lavaei, M. Khaled, S. Soudjani, and M. Zamani. AMYTISS: Parallelized automated controller synthesis for large-scale stochastic systems. In *Proc. 32nd International Conference on Computer Aided Verification (CAV)*, LNCS. Springer, 2020.

[53] A. Lavaei, S. Soudjani, A. Abate, and M. Zamani. Automated verification and synthesis of stochastic hybrid systems: A survey. *Automatica*, 2022.

[54] A. Lavaei, S. Soudjani, and M. Zamani. Compositional abstraction-based synthesis for networks of stochastic switched systems. *Automatica*, 114, 2020.

[55] A. Lavaei, S. Soudjani, and M. Zamani. Compositional abstraction of large-scale stochastic systems: A relaxed dissipativity approach. *Nonlinear Analysis: Hybrid Systems*, 36, 2020.

[56] A. Lavaei and M. Zamani. From dissipativity theory to compositional synthesis of large-scale stochastic switched systems. *IEEE Transactions on Automatic Control*, 2022.

[57] Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. Scalable verification of markov decision processes. In Carlos Canal and Akram Idani, editors, *Software Engineering and Formal Methods - SEFM 2014 Collocated Workshops: HOFM, SAFOME, OpenCert, MoKMaSD, WS-FMDS, Grenoble, France, September 1-2, 2014, Revised Selected Papers*, volume 8938 of *Lecture Notes in Computer Science*, pages 350–362. Springer, 2014.

[58] H. Ma and H.A.P. Blom. Interacting particle system based estimation of reach probability of general stochastic hybrid systems, preprint submitted 23rd may 2022.

[59] H. Ma and H.A.P. Blom. Random assignment vs. fixed assignment in multilevel importance splitting for estimating stochastic reach probabilities. *Methodology and Computing in Applied Probability*, pages 1–26, 2022.

[60] Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Symbolic control for stochastic systems via parity games. *arXiv preprint arXiv:2101.00834*, 2021.

[61] Rupak Majumdar, Kaushik Mallik, Anne-Kathrin Schmuck, and Sadegh Soudjani. Symbolic qualitative control for stochastic systems via finite parity games. *IFAC-PapersOnLine*, 54(5):127–132, 2021.

[62] Rupak Majumdar, Kaushik Mallik, and Sadegh Soudjani. Symbolic controller synthesis for Büchi specifications on stochastic systems. In *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, HSCC '20, New York, NY, USA, 2020. Association for Computing Machinery.

[63] Mathis Niehage, Arnd Hartmanns, and Anne Remke. Learning optimal decisions for stochastic hybrid systems. In S. Arun-Kumar, Dominique Méry, Indranil Saha, and Lijun Zhang, editors, *MEMOCODE '21: 19th ACM-IEEE International Conference on Formal Methods and Models for System Design, Virtual Event, China, November 20 - 22, 2021*, pages 44–55. ACM, 2021.

[64] 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.

[65] 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, pages 157–169. IEEE, 2018.

[66] 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, pages 186–191. ACM, 2017.

[67] 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*, pages 475–486. IEEE, 2017.

[68] Stefan Schupp, Erika Ábrahám, Ibtissem Ben Makhlouf, and Stefan Kowalewski. HyPro: A C++ Library of State Set Representations for Hybrid Systems Reachability Analysis. In Clark Barrett, Misty Davies, and Temesghen Kahsai, editors, *NASA Formal Methods*, volume 10227, pages 288–294. Springer, 2017.

[69] S. Soudjani and A. Abate. Adaptive and sequential gridding procedures for the abstraction and verification of stochastic processes. *SIAM Journal on Applied Dynamical Systems*, 12(2):921–956, 2013.

[70] S. Soudjani and A. Abate. Probabilistic reach-avoid computation for partially degenerate stochastic processes. *IEEE Transactions on Automatic Control*, 59(2):528–534, Feb 2014.

[71] S. Soudjani and A. Abate. Quantitative approximation of the probability distribution of a Markov process by formal abstractions. *Logical Methods in Computer Science*, 11(3):1–29, 2015. arXiv:1504.00039.

[72] Sadegh Soudjani, Caspar Gevaerts, and Alessandro Abate. FAUST^2: Formal Abstractions of Uncountable-STate STochastic processes. In *TACAS*, volume 15, pages 272–286, 2015.

[73] I. Tkachev and A. Abate. Characterization and computation of infinite horizon specifications over markov processes. *Theoretical Computer Science*, 515:1–18, 2014.

[74] I. Tkachev, A. Mereacre, J.-P. Katoen, and A. Abate. Quantitative model checking of controlled discrete-time markov processes. *Information and Computation*, 253(1):1–35, 2017.

[75] P. Turati, N. Pedroni, and E. Zio. Advanced RESTART method for the estimation of the probability of failure of highly reliable hybrid dynamic systems. *Reliability Engineering & System Safety*, 154:117–126, 2016.

[76] B. C. van Huijgevoort and S. Haesaert. Temporal logic control of nonlinear stochastic systems using a piecewise-affine abstraction. Technical report, 2022. https://www.sofiehaesaert.com/assets/Research/PWA_abstractions.pdf.

[77] Birgit C van Huijgevoort and Sofie Haesaert. Similarity quantification for linear stochastic systems: A coupling compensator approach. *Automatica*, 144:110476, 2022.