EPiC Series in Computing

Volume 80, 2021, Pages 55–89

8th International Workshop on Applied Verification of Continuous and Hybrid Systems (ARCH21)

ARCH-COMP21 Category Report: Stochastic Models

Alessandro Abate[1], Henk Blom[2], Marc Bouissou[3], Nathalie Cauchi[1], Hassane Chraibi[3], Joanna Delicaris[4], Sofie Haesaert[5], Arnd Hartmanns[6], Mahmoud Khaled[7], Abolfazl Lavaei[8], Hao Ma[2], Kaushik Mallik[9], Mathis Niehage[4], Anne Remke[4], Stefan Schupp[10], Fedor Shmarov[11], Sadegh Soudjani[12], Adam Thorpe[13], Vlad Turcuman[1], and Paolo Zuliani[12]

[1] University of Oxford, Oxford, UK
[2] Delft University of Technology, Delft, The Netherlands
[3] R&D Division of Electricité de France (EDF), France
[4] University of Münster, Germany
[5] TU Eindhoven, Eindhoven, The Netherlands
[6] University of Twente, Enschede, The Netherlands
[7] Technical University of Munich, Germany
[8] ETH Zurich, Switzerland
[9] Max Planck Institute for Software Systems, Germany
[10] TU Wien, Vienna, Austria
[11] University of Manchester, Manchester, UK
[12] Newcastle University, Newcastle upon Tyne, UK
[13] University of New Mexico, USA

Abstract

This report presents the results of a friendly competition for formal verification and policy synthesis of stochastic models. It also introduces new benchmarks within this category, and recommends next steps for this category towards next year's edition of the competition. The friendly competition took place as part of the workshop Applied Verification for Continuous and Hybrid Systems (ARCH) in Spring/Summer 2021.

1 Introduction

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 obtained during the ARCH Friendly Competition 2021 in the group *stochastic models*. A recent survey of many of the discussed techniques can be found at [60], and provides the theoretical underpinnings for much of the computational results presented here. The benchmark collection has been extended by 3 interesting benchmarks which exhibit novel challenges for tools participating in this category.

In this year's edition overall 11 tools and frameworks participated in the evaluation of earlier and new benchmarks. This year the following tools and frameworks participate (in alphabetical order): AMYTISS, FIGARO workbench, hpnmg, HYPEG, Mascot-SDS, the Modest Toolset, ProbReach, PyCATSHOO, SDCPN & IPS, SReachTools, StocHy, and SysCore. Similar to last year, 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, which include a discussion of the individual models syntax and semantics, and a presentation of the specifications of interest is presented in Section 4. 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.

AMYTISS AMYTISS is a software tool for designing correct-by-construction controllers of stochastic discrete-time 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 [79, 80, 81, 82]. AMYTISS is implemented as a kernel on top of the acceleration ecosystem pFaces [49]. It implements parallel algorithms to (1) build finite MDPs as finite abstractions of given original stochastic discrete-time systems, and (2) synthesize controllers for the constructed finite MDPs satisfying bounded-time safety specifications and reach-avoid specifications. The underlying computation parts are similar to the ones used in FAUST^2 and StocHy, and are used for compositional computations [83, 55, 56, 57, 58, 59].

AMYTISS significantly improves performances w.r.t. the computation time by parallel execution in different heterogeneous computing platforms including CPUs, GPUs and hardware accelerators (*e.g.,* FPGA). In addition, AMYTISS proposes a technique to reduce the required memory for computing finite MDPs as *on-the-fly abstractions* (OFA). In the OFA technique, computing and storing the probability transition matrix are skipped. Instead the required entries of the finite MDP are on-the-fly computed as they are needed for the synthesis part via the standard dynamic programming. This technique impressively reduces the required memory but at the cost of repeated computation of their entries in each time step from 1 to a finite-time horizon T_d. This gives the user an additional control over the trade-off between the computation time and memory usage. The tool is available at `https://github.com/mkhaled87/pFaces-AMYTISS`.

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, 13] 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 [51].

hpnmg The tool hpnmg [44] is a model checker for Hybrid Petri nets with an arbitrary but finite number of general transition firings against specifications formulated in STL [46]. 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 [45], transformation to a geometric representation as convex polytopes [47], 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.

The tool is implemented in C++ and relies on the library HyPro [74] for efficient geometric operations on convex polytopes, as well as on the GNU Scientific Library (GSL) for multi-dimensional integration using Monte Carlo integration [61]. Different approaches to multi-dimensional integration have been compared w.r.t. scalability in [43].

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

HYPEG The Java-based library HYPEG [71] is a dedicated statistical simulator for hybrid Petri nets with general transitions (HPnGs) [35], which combine discrete and continuous components with a possibly large number of random variables, whose stochastic behavior

follows arbitrary probability distributions. HYPEG uses time-bounded discrete-event simulation and well-known statistical model checking techniques to verify complex properties, including time-bounded reachability [72]. These techniques comprise several hypothesis tests as well as different approaches for the computation of confidence intervals. In the latest version of HYPEG, continuous behavior that can be expressed by systems of ordinary differential equations can be simulated using an approximative approach [70, 68], whereas piecewise-linear continuous behavior is simulated without approximation.

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

Mascot-SDS Mascot-SDS [66, 65, 64] is an open-source tool for synthesizing formally verified controllers for almost sure satisfaction (i.e. satisfaction with probability 1) of infinite-horizon specifications for discrete-time dynamical systems in the presence of stochastic perturbations. From version 1.1, Mascot-SDS accepts specifications expressed using either LTL formulas or Rabin automata, defined over polytopic predicates over the state space. The tool can compute both over- and under-approximations of the optimal controller domain for the almost sure satisfaction of the specification. It is written in C++, and is an extension of Mascot [42]. The suffix SDS stands for Stochastic Dynamical Systems.

Mascot-SDS is available at https://gitlab.mpi-sws.org/kmallik/mascot-sds.

The Modest Toolset The Modest Toolset [41] supports the modelling and analysis of hybrid, real-time, distributed and stochastic systems. At its core is the model of networks of stochastic hybrid automata (SHA) [40], which combine nondeterministic choices, continuous system dynamics, stochastic decisions and timing, and real-time behaviour, including nondeterministic delays. The Modest Toolset is a modular framework, supporting as input the high-level Modest modelling language [8, 40] and the Jani specification [16], and providing a variety of analysis backends for various special cases of SHA. In particular, the modes discrete-event simulator [15] supports SHA without nondeterminism and linear dynamics. It includes a highly-automated rare event simulation engine based on importance splitting [14], and provides statistical estimates with configurable error and confidence levels. The prohver tool [40] model-checks SHA with linear differential equations and inclusions, combining abstraction of continuous probability distributions with a non-stochastic hybrid automata reachability analysis (using PHAVer [31] as backend). It delivers guaranteed upper bounds on (time-bounded) reachability probabilities.

The Modest Toolset can can be obtained from https://www.modestchecker.net/.

ProbReach ProbReach [76] provides a set of algorithms for computing probabilistic bounded reachability in *parametric* stochastic hybrid systems (PSHS) with nonlinear continuous dynamics (i.e., defined by nonlinear ODEs). The parameters can be random (continuous and discrete) and/or nondeterministic. ProbReach features a formal approach [78] for computing numerically sound probability intervals that are formally guaranteed to contain the exact value of the bounded reachability probability. For PSHSs containing random parameters only, the size of such interval can be made arbitrarily small. Also, ProbReach implements Monte Carlo algorithms [77] for computing confidence intervals for the bounded reachability probability with rigorous (i.e., numerically sound) sampling. ProbReach uses the Probabilistic Delta-ReacHability (PDRH) format for encoding PSHSs, and it is available at https://github.com/dreal/probreach.

SReachTools SReachTools [95] is an open-source, repeatability-evaluated, MATLAB toolbox for tackling the problem of stochastic reachability of a target tube [99]. This problem subsumes terminal hitting-time stochastic reach-avoid and stochastic viability problems (guaranteeing

safety in stochastic systems) [85, 4]. SReachTools handles linear, discrete-time, continuous-state dynamical systems with additive stochastic disturbance. The dynamics and the safety constraints can be time-varying, and the disturbance may be Gaussian or non-Gaussian. It relies on approaches drawn from convex optimization, Fourier transforms, scenario-based optimization, and computational geometry for a grid-free and scalable computation of the stochastic reach sets as well as controller (open-loop, affine feedback, and set-based) synthesis [34, 96, 97, 100, 73]. The SReachTools Kernel Module [89] is the most recent addition to the toolbox, which implements a collection of nonparametric data-driven algorithms for stochastic reachability based on kernel methods. These algorithms leverage techniques from functional analysis and statistical learning theory and can handle discrete-time linear and nonlinear dynamical systems with arbitrary stochastic disturbances [86, 90].

SReachTools is available at https://sreachtools.github.io. The code for solving the benchmarks presented in this report is available as a Code Ocean capsule at https://doi.org/10.24433/CO.5339956.v1 and the Kernel Module code is available at https://doi.org/10.24433/CO.3853882.v1.

PyCATSHOO The PyCATSHOO [23] 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 [22]. 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.

StocHy StocHy [19] is a software tool for the quantitative analysis of discrete-time *stochastic hybrid systems* (SHS) accepts a high-level description of stochastic models and constructs an equivalent SHS model. In comparison with the other tools in the stochastic modelling category, StocHy is the only tool that provides exact (i.e. not via statistical means) errors/guarantees on

the obtained solution [20, 36, 53]. The tool enables users to (i) simulate the SHS evolution over a given time horizon; and to (ii) to automatically construct formal abstractions of the SHS using either abstractions taking the form of Markov Decision Processes (MDP) or grounded on interval MDPs (IMDP) [20, 53]. The abstractions are then employed for (ii) formal verification or (iii) control (policy, strategy) synthesis. StocHy allows for modular modelling, and has separate simulation, verification and synthesis engines, which are implemented as independent libraries. The tool is implemented in C++ and employs manipulations based on vector calculus,sparse matrices, symbolic construction of probabilistic kernels, and multi-threading. It computes the transition probabilities for the states of the IMDP in parallel using a bag-of-tasks approach. In order to compute the probabilities, StocHy has a set number of threads which take states and compute the transitions repeatedly, without waiting for other threads to be done.

The tool can be obtained from https://github.com/natchi92/stochy.

SysCore SysCore is developed for formal verification and 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, 37, 75, 39] 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 compare 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.

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) [17]. GSHS incorporates Brownian motion in continuous-time Piecewise Deterministic Markov Processes [25]. Because a direct specification of a large GSHS model does not work, the framework of Stochastically and Dynamically Coloured Petri Nets (SDCPN) [27, 28, 29, 30] 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 [21, 6, 7, 63]. The SDCPN & IPS framework is applied to the Heated Tank benchmark.

3 Established benchmarks, revisited

Benchmarks already established allow to visualise the development progress that a tool makes during its life-cycle. Therefore, the collection of benchmarks used for evaluation does not only consider new benchmarks but also re-uses existing benchmarks to allow tool developers

to improve their results from previous years. In this section we give a short description of established benchmarks that have been used in the evaluation with references to their original sources (and previous reports in ARCH) for further details.

3.1 Automated Anesthesia

The automated anesthesia delivery system benchmark is a stochastic hybrid model which models inputs from an anaesthesiologist within a safe and automated delivery system. The model description and specifications are described in ARCH 2018 [3].

3.2 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 [3] and [18].

3.3 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) [25]. This made the Heated Tank benchmark a logical candidate for inclusion in the set of ARCH stochastic models [3, 1, 2].

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. [24, 94], 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 [3], version 4 has been selected as most suitable rare event estimation benchmark. In [1] 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.

Heated Tank version 4.0 has been evaluated by the tools modes [3, 2] and HYPEG and by the method SDCPN&IPS [1]. In [1] 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.

3.4 Water Sewage Facility

The water sewage benchmark models a water sewage treatment plant situated in Enschede as a hybrid Petri net with general transitions (HPnG). It has been proposed and thoroughly

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

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

analyzed in [32] and was already included in ARCH2019 [1] and results have been computed during ARCH2020 [2].

The report from 2019 features two different failure scenarios for the water sewage facility [33]. While in 2020 results have been computed for both failure scenarios, we will focus on extension A in this year and adapt the previously evaluated *survivability property* into a *rare-event property*. This means that instead of estimating the probability that the system recovers in time, we now compute the probability that the buffer in front of the water sewage facility overflows: $\varphi_{A_{\text{rare}}} = (m_{P_{\text{n}}} = 0)\ \mathcal{U}^{[0,30]}\ (x_{P_0}\ \geq 0.1)$. Furthermore, the initial parameter setting for place P_1 has been changed to zero to further reduce the probability of overflow.

To make the benchmark accessible for tools that operate on stochastic hybrid automata, the hybrid Petri net model has been transformed into a hybrid automaton with stochastic resets, in [2]. This year, the stochastic hybrid automaton has also been modeled from scratch in the Modest language; for ProbReach, the model required the introduction of another mode (courtesy of Arnd Hartmanns) with respect to the 2020 model [2].

3.5 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 $\square\lozenge B$, the following modified dynamical system can be used together with a reachability specification that gives an upper-bound for probability of satisfying $\square A \wedge \square\lozenge B$. 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 $\lozenge\phi_2$ 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 $\square A \wedge \square\lozenge B$.*

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

$$\mathcal{P}_i = \{X \in \mathbb{R}^2 \,|\, AX \leq B_i\}, \quad i \in \{1, 2, 3\},$$

in the two-dimensional space. The first two polytopes $\mathcal{P}_1$ and $\mathcal{P}_2$ specify a region around the limit cycle of the system. The dynamics of the system in continuous time is as follows

$$\begin{aligned} dx_1 &= x_2 dt + dW_1 \\ dx_2 &= (-x_1 + (1 - x_1^2)x_2)dt + dW_2, \end{aligned} \tag{4}$$

with W_1 and W_2 being standard Brownian motion.

Problem 5 (Rare event computation)**.** *Compute the probability that the trajectory goes outside of $\mathcal{P}_1$ and $\mathcal{P}_2$ around the limit cycle in the time internal $[0, T]$ after entering the polytope $\mathcal{P}_3$:*

$$Prob\left(\exists t_1, t_2 \in [0, T],\, t_2 \geq t_1 \,\wedge\, X(t_1) \in \mathcal{P}_3 \,\wedge\, (X(t_2) \notin \mathcal{P}_1 \vee X(t_2) \notin \mathcal{P}_2)\right). \tag{5}$$

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

$$A = \begin{bmatrix} +\alpha_1 & -1 \\ -\alpha_2 & +1 \\ -\alpha_1 & +1 \\ +\alpha_2 & -1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} -\beta_1 \\ +\beta_2 \\ -\beta_1 \\ +\beta_2 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -\gamma_1 \\ +\gamma_2 \\ -\gamma_1 \\ +\gamma_2 \end{bmatrix}, \quad B_3 = (B_1 + B_2)/2,$$

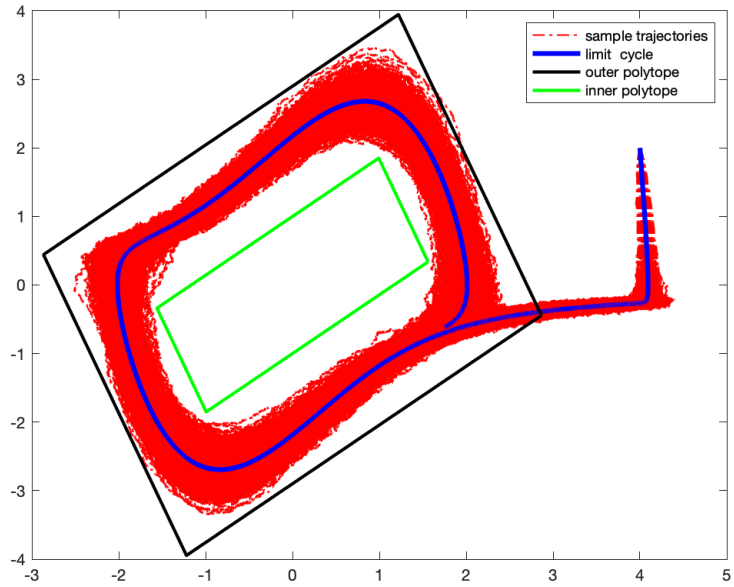

Figure 1: Sample trajectories of the stochastic Van der Pol Oscillator starting from initial state $X_0 = [4, 2]^T$. Most of the trajectories remain between the black polytope $\mathcal{P}_1$ and the green polytope $\mathcal{P}_2$.

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 1 together with polytopes $\mathcal{P}_1$ (in black) and $\mathcal{P}_2$ (in green) and the limit cycle for the deterministic version of the system (in blue). Applying a standard Monte Carlo approach to this problem with $10,000$ trajectories gives the estimate 0.0011 for the probability in (5). 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).

3.6 Integrator-Chain (for scalability comparison)

This benchmark is used as a measure of tool scalability and is described in ARCH 2020 [2].

3.7 7-Dimensional BMW 320i

This benchmark corresponds to that presented in [2].

4 New Benchmarks

Tool development strongly profits from large sets of benchmarks for evaluation of the implemented approach. Additionally to the already existing benchmarks which have been proposed in the last years [2, 1, 3], in this section we present a new benchmark which has been proposed this

year for evaluation and introduce new challenges that are not currently handled by the existing benchmarks.

4.1 Patrol Robot

Consider the controller synthesis problem for a robotic vehicle that is required to fulfill an assume-guarantee style specification. Suppose the vehicle is placed in a work-space with two rooms which are separated by a door, and needs to infinitely often visit both rooms if the door opens infinitely often (and remains open for "long enough" for the vehicle to be able to pass, where "long enough" will be formalized later). The dynamics of the vehicle is modeled using the perturbed sampled-time model of the 3-d Dubins vehicle having the state variables x_1, x_2, and x_3, representing respectively its position along the X-axis in m, position along the Y-axis in m, and the angle of steering wheel in rad, and the control input variable u representing the change in steering angle in rad/s. We also additionally include a boolean state variable x_4 that represents whether the door is open or not (1 for open and 0 for closed). For $u \neq 0$, the dynamics is given as:

$$\begin{aligned} x_1(k+1) &= x_1(k) + \frac{V}{u}\sin(x_3(k)+u\tau) - \frac{V}{u}\sin(x_3(k)) + w_1(k) \\ x_2(k+1) &= x_2(k) - \frac{V}{u}\cos(x_3(k)+u\tau) + \frac{V}{u}\cos(x_3(k)) + w_2(k) \\ x_3(k+1) &= x_3(k) + u\tau + w_3(k) \\ x_4(k+1) &\in \{0,1\}, \end{aligned} \tag{6}$$

and when $u = 0$, the dynamics can be obtained as the limit of the above. The velocity V is fixed to 1 m/s, and the sampling time τ is chosen to be 0.1 s. Let the stochastic noise samples $(\varsigma_1(k), \varsigma_2(k), \varsigma_3(k))$ be drawn from a distribution with support $D = [-0.06, 0.06] \times [-0.06, 0.06] \times [-0.06, 0.06]$, and the range of the state and the input variables be as follows: $x_1 \in [-0.6, 0.96]$, $x_2 \in [-1.2, 1.98]$, $x_3 \in [-\pi, \pi]$, and $u \in [-20, 20]$.

The state space of the system is decorated with the predicates loc_A, loc_B, and $door$, which are given as the inverse projections of the following respective rectangles in the two-dimensional space spanned by x_1 and x_2: $[0.2, 0.8] \times [1.2, 1.8]$, $[-0.4, 0.2] \times [-0.9, -0.3]$, and $[-0.6, 0.96] \times [0.65, 0.7]$.

The specification is given as:

$$\begin{aligned} \Box\Diamond(x_4 = 1) \wedge \Box\left((x_4 = 1) \rightarrow (x_4 = 1)\,\mathcal{U}\; loc_B\right) \rightarrow \\ (\Box\Diamond loc_A \wedge \Box\Diamond loc_B \wedge \Box\neg(door \wedge (x_4 = 0))\,. \end{aligned} \tag{7}$$

Intuitively, the left side of the above implication is an assumption on the environment which states that the door opens ($x_4 = 1$) infinitely often and whenever it opens, it remains open until the vehicle visits loc_B the next time (giving the vehicle long enough time to pass through the door). The right side of the implication is the guarantee required from the vehicle which states that the vehicle needs to visit both loc_A and loc_B infinitely often, and should always avoid collision with the closed door.

Problem 6. *Compute the set of initial states and the witness controller for the system* (6)*, for which the probability of satisfying the specification* (7) *equals to* 1*.*

4.2 Control synthesis version of Heated Tank

Consider an unheated tank inspired by the heated tank already included in the benchmarks where the units are repaired by a repairman who can be scheduled. We model the tank with

one outflow valve and two pumps adding water to the tank. At any time, each pump is in state "on", "off", "silently failed", or "failed", where the rates of the pump in the failed states equal the rates in the "off" state. The valve is in state "on", "off", "silently stuck on", "stuck on", "silently stuck off", or "stuck off". We propose two versions, (a) one with piecewise linear and (b) one with nonlinear continuous behaviour inspired by Toricelli's law. Hence, the water level h in the tank is modelled by one of the following two differential equations:

(a) $\dot{h} = p_1 + p_2 - v$

(b) $\dot{h} = p_1 + p_2 - \frac{a_v}{a_t} \cdot \sqrt{2 \cdot G \cdot h}$

The flow rate of pump 1 is p_1 which equals constant *PUMP1* if the pump is "on" and 0 otherwise. Pump 2 behaves similarly with constant *PUMP2*. In version (a), the flow rate of the valve is $v = VALVE$ if the valve is "open" or "silently stuck open" and $v = 0$ otherwise. In contrast, in version (b), the flow rate of the valve is given by Toricelli's law with the following constants:

a_v the diameter of the valve,

a_t the diameter of the tank, and

G the gravitational acceleration on Earth which we approximate with 9.81.

The tank has three constants as thresholds: H_{HIGH}, H_{MID}, and H_{LOW} with $H_{HIGH} > H_{MID} > H_{LOW}$. Initially the tank height equals H_{MID}.

Fixed controller. The controller is in one of four modes: "Normal", "Degraded", "Increase", or "Decrease", at any time. Initially, it enters "Normal" mode. Upon entering a new mode, the following switching actions are performed:

- Into mode "Normal": turn on pump 1, turn off pump 2, open the valve.
- Into mode "Degraded": turn off pump 1, turn on pump 2, open the valve.
- Into mode "Increase": turn on pump 1, turn on pump 2, close the valve.
- Into mode "Decrease": turn off pump 1, turn off pump 2, open the valve.

If a pump is in state "silently failed" or "failed", or the valve is in state "silently stuck on", "stuck on", "silently stuck off", or "stuck off", then a switching action does not change its state as it would normally (see below).

Immediately when h reaches H_{HIGH} and the current mode is "Normal" or "Degraded", the controller enters mode "Decrease". Immediately when h reaches H_{LOW} and the current mode is "Normal" or "Degraded", the controller enters mode "Increase". Immediately when h reaches H_{MID} and the current mode is "Increase" or "Decrease", the controller enters mode "Degraded" if pump 1 is "silently failed" or "failed", and mode "Normal" otherwise.

An attempt to switch on a pump that is in state "silently failed" changes its state to "failed". An attempt to open or close the valve when it is "silently stuck open" or "silently stuck closed" changes its state: from "silently stuck open" to "stuck open" and from "silently stuck closed" to "stuck closed", respectively. That is, an attempt to switch a silently defective pump or valve reveals the defect. The rule for H_{MID} above already accounts for this.

Failures. The pumps can fail in two ways: When a pump is not used, it slowly decays until it silently fails; the deterioration process is completely reset when it is turned on before failure. All the usage time of a pump counts towards its wear; after some time of use, it will (non-silently) fail. Thus we have the following random processes:

- **Pump decay:** When a pump changes its state to "off", the time until it silently fails follows an exponential distribution with rate $PUMP_{DECAY}$.
- **Pump wear:** From the initial state, and following a repair, the accumulated time in state "on" until the pump fails follows a Weibull distribution with shape $k = PUMP_{WEAR_k}$ and $\lambda = PUMP_{WEAR_\lambda}$. (To obtain a Markovian variant of the model, use the exponential distribution with the same mean.)
- **Valve decay:** When the valve changes its state to "on" or "off", the time until it silently fails (from "on" to "silently stuck on" and from "off" to "silently stuck off") follows an exponential distribution with rate $VALVE_{DECAY}$.

When the controller is in mode "Normal" and pump 1 changes its state to "failed", then the controller immediately enters "Degraded" mode.

Repairs (to be synthesized). A repairman repairs at most one pump or valve at the same time. We consider two classes of repair policies:

- **non-preemptive:** once a repair has started, it will be completed without interruption;
- **preemptive:** if, during a repair, another pump or valve fails, the repairman may decide to repair that one instead.

The valve is easier to repair than a pump. As specified above, repairing a valve or pump completely resets its decay and wear. The effect of a repair on the repaired component's state depends on the mode of the controller:

- In mode "Normal":
  - pump 1 cannot be failed (then we are in mode "Degraded");
  - if pump 2 is repaired, turn off pump 2;
  - if the valve is repaired, open the valve.
- In mode "Degraded":
  - if pump 1 is repaired, switch to mode "Normal";
  - if pump 2 is repaired, turn on pump 2;
  - if the valve is repaired, open the valve.
- In mode "Increase":
  - if pump 1 is repaired, turn on pump 1;
  - if pump 2 is repaired, turn on pump 2;
  - if the valve is repaired, close the valve.
- In mode "Decrease":

  - if pump 1 is repaired, turn off pump 1;
  - if pump 2 is repaired, turn off pump 2;
  - if the valve is repaired, open the valve.

The times to repair are as follows:

- For a pump in state "failed": $PUMP_{PREP}$ time units to prepare the repair, followed by an amount of time following a continuous uniform distribution over the interval $[PUMP_{REP_l}, PUMP_{REP_u}]$ for the actual repair. (To obtain a Markovian model, use the exponential distribution with the same mean.)

- For a valve in state "stuck open" or "stuck closed": $VALVE_{PREP}$ time units to prepare the repair, followed by an amount of time following a continuous uniform distribution over the interval $[VALVE_{REP_l}, VALVE_{REP_u}]$ for the actual repair. (For a Markovian model, use the exponential distribution with the same mean.)

In case of preemption, the time to prepare the repair is incurred again every time the repairman switches back to an already-started repair; after that, the remaining time for the actual repair is needed. In the non-preemptive case, if two more components fail while one is being repaired, the subsequent choice of which to repair next is not specified and subject to control synthesis. Additionally, in the preemptive case, if one more component fails while another one is being repaired, then the immediate decision of whether to switch to that one is not specified and subject to control synthesis. As a baseline for the synthesis performance, we can consider a uniformly random policy: Whenever there is a decision between repairing $n > 1$ different components, use a discrete uniform distribution to select one.

Synthesis considerations. Realistically, the repair policy should not be aware of silent defects. Thus we have a partially observable discrete state. If the model is implemented with a sample-and-wait semantics, then it is important to distinguish between prophetic and non-prophetic policies. The former know the remaining times to failures and repairs; the latter do not.

Optimisation goal. Minimize the probability for the tank to become empty (because then it would surely overheat) within T time units: $P_{min}(\Diamond^{time \leq T} h = 0)$

Summary of variants. Within the description several variants were introduced which are summarized in the following:

- linear or Toricelli;
- general continuous random distributions or Markovian;
- preemptive or non-preemptive repairs;
- complete information or partial observability;
- prophetic or non-prophetic policies.

4.3 Geometric Brownian motion

The Geometric Brownian motion benchmark is taken from [52]. The SDE of Geometric Brownian motion satisfies:

$$dX_t = (\mu + \frac{\sigma^2}{2})X_t dt + \sigma X_t dW_t \tag{8}$$

where$X_0 \geq 1$, $\mu > 0$ and $\sigma > 0$. We want to estimate the probability $\mathbb{P}\{\tau < T\}$ with $\tau \triangleq \inf\{t > 0 : X_t \geq L\}$. The proposed Geometric Brownian motion benchmark has parameter values: $X_0 = 1$, $\mu = 1$, $\sigma = 1$, level $L = 1717.25$ and T=1s. Following [93, 48], we can use the following equation to evaluate reach probabilities:

$$\gamma = \mathbb{P}(\tau < T) = \int_0^T \frac{\ln(L/X_0)}{\sqrt{2\pi\sigma^2 t^3}} \exp\left\{\frac{-[\ln(L/X_0) - \mu t]^2}{2\sigma^2 t}\right\} dt \tag{9}$$

We have used this equation to determine the value for the final level $L = L_{10} = 1717.25$, and also use this equation to determine values for 9 intermediate levels L_k, $k = 1, ..., 9,$, such that the conditional hit probabilities between successive levels are equal to 1/10. Table 2 shows the resulting L_k level values for k=1,2,..., as well as the analytical γ results for these levels.

The right hand column in Table 2 also show estimated $\bar{\gamma}_{MC}$ results that have been obtained through conducting a straightforward Monte Carlo (MC) simulation using 10000 runs and numerical integration time step $\Delta = 2 \times 10^{-3} s$. These $\bar{\gamma}_{MC}$ results show that straightforward MC simulation based estimation of γ fails to work beyond $k = 4$.

Table 2: Analytical γ for geometric Brownian motion benchmark, with $X_0 = 1$, $\mu = 1, \sigma = 1$, at level $L = L_{10} = 1717.25$ and $T = 1s$ and 9 intermediate levels.

k	L_k	γ
1	12.27	0.09998
2	33.038	1.000×10^{-2}
3	69.09	1.000×10^{-3}
4	127.45	1.001×10^{-4}
5	217.5	1.000×10^{-5}
6	351.445	1.000×10^{-6}
7	545.14	1.000×10^{-7}
8	818.935	1.000×10^{-8}
9	1198.75	1.000×10^{-9}
10	1717.25	1.000×10^{-10}

5 Friendly Competition – Setup and Outcomes

The results observed from the execution of the previously described benchmarks with the participating tools are presented in this section, we present an overview in Table 3.

Our efforts in providing a more detailed classification of benchmarks used in the competition which were started last year have been continued this year. All novel benchmarks which were presented this year have been classified according to the criteria specified last year (see Table 4).

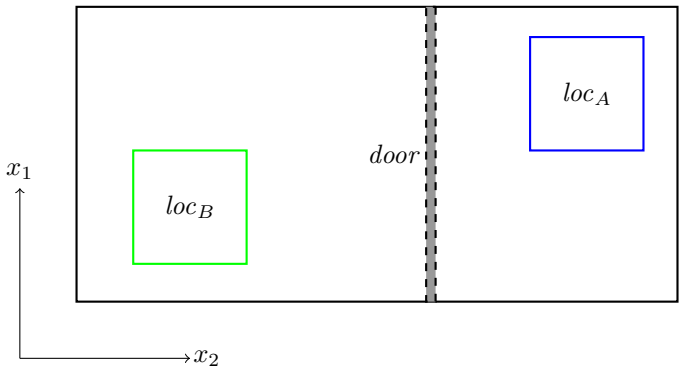

Figure 2: Visualization of the state space predicates of the service robot benchmark.

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).

Tool	Benchmarks								
	AS	BA	HT	WS	VP	IC	AV	PR	GB
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					
Mascot-SDS					2020			2021	
modes			2018	2020					
ProbReach				2020					
prohver			2020	2020					
SDCPN&IPS			2019						2021
SysCore	2021	2021							
Figaro			2021						
PyCATSHOO			2021						

An overview on the evaluation results obtained this year on old and new benchmarks is given in Table 3.

5.1 Centralized execution

Following the goal of a centralized execution which is aimed for in all categories of the ARCH-Competition, we encouraged tool developers to hand in Docker containers for a repeatability evaluation to the repeatability committee at ARCH See https://gitlab.com/goranf/ARCH-COMP/tree/master/2021/SM for more details on the repeatability evaluation and the provided repeatability evaluation packages by the tool developers.

Nonetheless, the running times presented here are resulting from the execution of the tools

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
Liveness/deadlock					✓			✓	
Prob. reachability	✓	✓	✓	✓		✓	✓		✓
Control synthesis	✓	✓				✓	✓	✓	
Min-max		✓					✓		
Time horizon	F	F	F	F	I	F	F	I	F
Type of control	S	M			Dr	Dr	M	M	
Time line	D	D	C	C	D	D	D	D	C
State space	C	H	H	H	C	C	C	H	C
Drift in ODE/SDE	pL	NL	NL	pL	NL	L	NL	NL	L
Noise in SDE	Fixed	Fixed			Fixed	Fixed	Fixed	Fixed	State
Noise: BM or i.i.d.	iid	iid			iid	iid	iid	iid	BM
Guards		✓	✓	✓			✓		
Rate spontaneous jumps	Fixed		State	Fixed		Fixed			
Size spontaneous jumps	Fixed		Fixed	Fixed		Fixed			
Environment		✓		✓			✓	✓	
Subsystems		✓	✓	✓					
Concurrency			✓	✓					
Synchronization			✓	✓					
Shared variables		✓		✓					
# discrete states		5	576	35				2	
# continuous variables	3	7	2	11	2	50	7	4	1
# model parameters	24	19	15	36	3	8	11	2	5

on the tool-maintainers' own machines.

5.2 Anesthesia benchmark results

Table 5 compares the performance of the tools based on their run time and the highest maximum stochastic reach probability starting from any state in the initial safe set for the Anesthesia benchmark. This benchmark defines a stochastic viability problem for a three-dimensional Gaussian-perturbed LTI system model.

Table 5: Run times and maximum reach probability on the Anesthesia benchmark.

Property	StocHy	AMYTISS
Run time on common CPU (sec)	0.402	0.288
Maximum reach probability	$\geq 0.99 \pm 0.02$	≈ 0.99

AMYTISS has the fastest run time, while producing among the highest stochastic reach probability along with StocHy and SReachTools. StocHy has gained a 13.39 speed-up improvement

with parellelisation of this case study, making it's overall run time comparable to AMYTISS.

StocHy computes the optimal control policy which maximises the probability of satisfaction given any initial condition within the input set and provides exact guarantees on the resulting solution. For this case study, we see that StocHy can keep the system safe with 0.99 probability with a maximum abstraction error of 0.02.

5.3 Building automation benchmark results

Table 6 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 3.2).

Table 6

Property	StocHy	AMYTISS
Case 1, 4-dimensional system		
Run time on common CPU(sec)	7.17	0.92
Maximum reach probability	$\geq 0.99 \pm 0.05$	≈ 0.99
Case 2, 7-dimensional system		
Run time on common CPU (sec)	335.876	12.5
Maximum reach probability	$\geq 0.8 \pm 0.23$	≈ 0.8

AMYTISS has the fastest run time for both cases with comparable maximum reach probabilities. StocHy has been run with the new parallelisation method enabled and when comparing with the results from [2], we note a significant improvement. Note, StocHy was not able to solve case 2 in [2] and not can run it in 335s without reducing it's computed maximum reach probabilities. This further highlights the new improvements.

Since SysCore is currently going under a substantial development of the techniques, we have applied the tool only 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 Maximum reach probability is equal to 0.9035 with the total run time 159.6 seconds. These values are obtained by setting $(\epsilon, \delta) = (0.2, 0.0161)$. These values are not reported in the above table as they are obtained with a different computation platform. The code is also not optimized for parallel computations. it is expected that an optimized version of the code would provide a much better result.

5.4 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 [3, 1]. Within ARCH2021 the objectives is 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 [1].

The differential equation for the temperature $x_{T,t}$ of the liquid satisfies:

$$dx_{T,t} = [q(\chi_{P1,t} + \chi_{P2,t})(T_{in} - x_{T,t}) + E_{in}]/(x_{H,t} + H_{ref}).dt \tag{10}$$

Table 7: Parameter values of Heated Tank versions 4.0 and 4.III

Parameter	Version 4.0 value	Version 4.III value
$\hat{\lambda}_{P1}$	$2.2831 \times 10^{-3}\ h^{-1}$	$2.2831 \times 10^{-3}\ h^{-1}$
$\hat{\lambda}_{P2}$	$2.8571 \times 10^{-3}\ h^{-1}$	$2.8571 \times 10^{-3}\ h^{-1}$
$\hat{\lambda}_{V}$	$1.5625 \times 10^{-3}\ h^{-1}$	$1.5625 \times 10^{-3}\ h^{-1}$
b_1	3.0295	3.0295
b_2	0.7578	0.7578
b_c	0	0.05756
b_d	0	0.2301
$\lambda_{\text{repair}} = 2\mu$	$0.2\ h^{-1}$	$1\ h^{-1}$
q	$0.6\ m/h$	$1.5\ m/h$
E_{in}	$1°Cm/h$	$23.88915°Cm/h$
T_{in}	$15°C$	$15°C$
T_{init}	$15.667°C$	$30.9261°C$
H_{Overflow}	$5\ m$	$10\ m$
H_{High}	$1\ m$	$8\ m$
H_{Init}	$0\ m$	$7\ m$
H_{Low}	$-1\ m$	$6\ m$
H_{Dryout}	$-5\ m$	$4\ m$
H_{ref}	$9\ m$	$0\ m$
t_0	$0\ h$	$0\ h$
t_{end}	$500\ h$	$1000\ h$
T_{Boil}	$100°C$	$100°C$

with $x_{H,t}$ the height of the liquid in the tank, $\chi_{P1,t}$ and $\chi_{P2,t}$ are $(0,1)$ -indicators if pump 1 and pump 2 deliver input flow q or not, T_{in} is the temperature of inflowing liquid, E_{in} the heat energy produced by the source, and H_{ref} a reference level in the tank. All parameter values are given in Table 7.

Pump1, Pump2 and Valve undergo independent failure transitions: On $\rightarrow$ Stuck-on, On $\rightarrow$ Stuck-off, Off $\rightarrow$ Stuck-on, Off $\rightarrow$ Stuck-off at a temperature $x_{T,t}$ dependent rate $\hat{\lambda}_U \cdot \alpha\left(x_{T,t}\right)$ $U \in \{P1, P2,\ V\}$, with temperature dependent factor $\alpha\left(x_{T,t}\right)$, e.g. [101]:

$$\alpha\left(x_{T,t}\right) = \left[b_1 \exp\left\{b_c\left(x_{T,t} - 20\right)\right\} + b_2 \exp\left\{-b_d\left(x_{T,t} - 20\right)\right\}\right] / \left(b_1 + b_2\right) \tag{11}$$

Heated Tank version 4.III has been evaluated by FIGARO, PyCATSHOO, SDCPN&MC and SDCPN&IPS. The results for Dry-out probability are given in Table 8. 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).

Table 8: P-Dryout for Heated Tank version 4.III: estimated by FIGARO, PyCATSHOO, SDCPN & MC and SDCPN & IPS

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 à 100,000 part.

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 [22] 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 [1]. 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 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.

5.5 Water sewage facility benchmark results

Within this year's competition, the water sewage facility benchmark has been evaluated by the analytical model checkers hpnmg and ProbReach and the statistical model checkers HYPEG and modes. Property $\varphi_{A_{\text{rare}}}$ has been model checked for the water sewage facility with extension A. The model has been parametrized for the evaluation taking into account the new initial value for P_1 (it is now 0). From the results, a selection of parameter sets are presented in this report.

Setup. For extension A, the mean of the random variable modeling the duration of heavy rain is set to 1.5h and the capacity of the community buffer P_c ranges between 30 and 42 (in 10^6 liters). These parameter combinations have been chosen in order to yield increasingly lower probabilities.

For hpnmg, statistical errors are caused by numerical methods used for multi-dimensional integration. Using Monte Carlo integration, specifically the VEGAS method [61], with default settings and a predefined number of samples leads to estimated errors smaller than 10^{-6}. We configured the Monte Carlo simulation-based tools—HYPEG and modes—to sample a number of runs sufficient to obtain a confidence level of 95 % with a confidence interval half-width of 10 % of the estimate.

Results. Table 9 presents the results and computation times for $\varphi_{A_{\text{rare}}}$. A large community buffer and a small mean for the duration of rain result in a very low probability of overflowing.

All tools except ProbReach are able to compute the probability that $\varphi_{A_{\text{rare}}}$ holds for varying values of P_c. For ProbReach, due to a bug in the software it was not possible to compute rigorous enclosures. Nevertheless, the bounds reported in Table 9 are numerically sound, in the sense that they are guaranteed to bound from above the true probability value.

For $P_c = 30$ all computed probabilities coincide such that at least the confidence interval overlap; modes' results appear to be lower than the others. For all larger values of P_c, hpnmg computes slightly larger probabilities than ProbReach, while the results of ProbReach and the different simulation tools match w.r.t. the computed confidence intervals. The statistical error for the results computed by hpnmg becomes smaller with the computed probability. We expect that this is due to the adaptive integration VEGAS, which uses importance sampling to concentrate evaluations of the integrand to regions where it is largest in magnitude [61]. Furthermore, for $P_c = 42$ the probabilities computed by modes MC and Restart exceed the result by ProbReach. Note that for larger values of P_c the confidence intervals computed for HYPEG were not able to match the accuracy achieved by modes due to performance issues.

Computation times. Comparing the simulative approaches, HYPEG requires significantly more time. This follows partly from the fact that HYPEG takes the original Petri net model as input and has to compute new rates for continuous places whenever a place in the model gets empty or full, c.f. *rate adaptation* in [2]. Furthermore, modes stops a simulation run as soon as the property is decided. In this case study, this corresponds to modes stopping a run as soon as the rain stops. The mean duration of rain is 1.5h and the maximum simulation time 30h, hence this optimization by modes should be quite influential on the computation times as well.

When using the Restart rare event simulation method, modes performs only slightly better than when using standard Monte Carlo simulation ("modes MC") for large community buffers as the estimated probability becomes smaller. As importance function, modes Restart uses a discretisation of the water level in the community buffer; this is evidently not good enough to

provide a significant benefit. In general, we found that simulation runs are rather short, leaving little room for an importance splitting method like Restart to gain from splitting the runs.

The analytical tools are in general faster than the simulations, which is to be expected since the model has only one random variable (the duration of heavy rain). Note that hpnmg has nearly constant computation times whereas ProbReach becomes faster with an increasing P_c (which in turn decreases the probability of satisfying $\varphi_{A_{\text{rare}}}$). The latter is due to the fact that it is computationally harder to evaluate rigorously the portions of the (random) parameter space over which $\varphi_{A_{\text{rare}}}$ is true with respect to those where it is false [78]. Since the probability becomes smaller precisely because a smaller portion of the parameter space satisfies the property, this speeds up ProbReach.

Table 9: Probabilities for $\varphi_{A_{\text{rare}}} = (m_{P_{\text{n}}} = 0)\ \mathcal{U}^{[0,30]}\ (x_{P_0} \geq 0.1)$, estimated error (for hpnmg) resp. confidence interval (for the other tools but ProbReach) and run time for the mean duration of raining $\mu = 1.5\,\text{h}$ and the capacity of the community buffer P_{c} for the water sewage facility benchmark with extension A. ProbReach's results are upper bounds of the probability of satisfying $\varphi_{A_{\text{rare}}}$. Note that the capacity of the community buffer P_{c} is given in 10^6 liters. Simulation parameters: Confidence level 95%, aim is to obtain an interval width of ±10% (not possible with HYPEG for 38, 42 since run time is too long).

Param.	Tools				
P_{c}	hpnmg	HYPEG	modes MC	modes Restart	ProbReach
30	$1.21 \cdot 10^{-3}$ $2.3 \cdot 10^{-7}$ 0.142 s	$1.19 \cdot 10^{-3}$ $\pm 1.0 \cdot 10^{-4}$ 309.5 s	$1.14 \cdot 10^{-3}$ ±10 % 1 s	$1.15 \cdot 10^{-3}$ ±10 % 1 s	$\leqslant 1.21 \cdot 10^{-3}$ 51 s
34	$1.54 \cdot 10^{-4}$ $3.1 \cdot 10^{-8}$ 0.133 s	$1.40 \cdot 10^{-4}$ $\pm 1.0 \cdot 10^{-5}$ 3314.5 s	$1.49 \cdot 10^{-4}$ ±10 % 6 s	$1.50 \cdot 10^{-4}$ ±10 % 6 s	$\leqslant 1.48 \cdot 10^{-4}$ 44 s
38	$1.44 \cdot 10^{-5}$ $2.8 \cdot 10^{-9}$ 0.134 s	$1.55 \cdot 10^{-5}$ $\pm 5.0 \cdot 10^{-6}$ 1452.8 s	$1.29 \cdot 10^{-5}$ ±10 % 67 s	$1.34 \cdot 10^{-5}$ ±10 % 59 s	$\leqslant 1.49 \cdot 10^{-5}$ 37 s
42	$9.73 \cdot 10^{-7}$ $1.1 \cdot 10^{-12}$ 0.106 s	$9.20 \cdot 10^{-7}$ $\pm 5.0 \cdot 10^{-7}$ 9426.2 s	$9.45 \cdot 10^{-7}$ ±10 % 865 s	$1.05 \cdot 10^{-6}$ ±10 % 705 s	$\leqslant 9.15 \cdot 10^{-7}$ 33 s

Outlook. After performing the computations for property $\varphi_{A_{\text{rare}}}$, we have observed that a significant part of the model, specifically the cleaning street, does not influence the computed probability. Hence, it would be of interest to formulate a property which relates e.g. to the digestion tank, to ensure that the complete state space of the model is constructed.

Platforms. HYPEG has been executed on a machine with an Intel Core i5-8250U (4×1.6–3.4 GHz) system and 16 GB memory. hpnmg was executed on an Intel i5-7200U processor with 2 CPU cores running on a frequency of 2.712 GHz and 16 GB memory. modes ran on a machine with an Intel i5-6600T processor with 4 physical CPU cores running at a frequency of 2.7–3.5 GHz, and 16 GB of memory. ProbReach was executed on a machine with 2x Intel Xeon 3.5 GHz E5-2637v3 processors (8 cores) and 32 GB of RAM (the timings reported in Table 9 are single-core).

5.6 Van der Pol Oscillator benchmark results

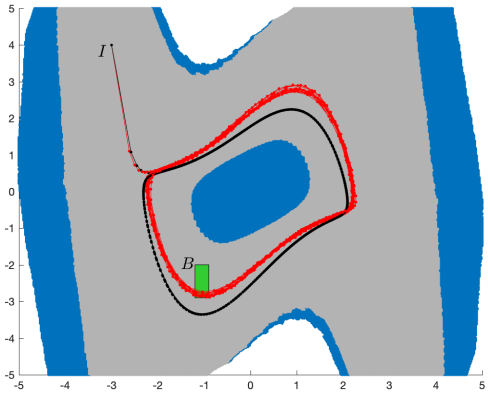

Figure 3: The Van der Pol oscillator example: The set B (green box) is the target that should be visited infinitely often. The under-approximation of the winning region is in grey. The over-approximation of the winning region is the union of blue and grey areas. I is the initial state for simulation. The trajectory with stochastic perturbation is shown in red, and the trajectory with a fixed deterministic perturbation that always misses the target is shown in black.

We have applied Mascot-SDS on this benchmark for solving Problem 1. An over- and under-approximation of the winning region is computed and plotted in Figure 3. Note that when the noise is treated as worst case, then there exists a deterministic value of the noise for which the oscillator trajectory never reaches the target B from all the initial states inside the domain, thus violating the specification. So the winning region is empty if the noise is treated as worst case. A trajectory with a fixed deterministic perturbation that misses the target all the time is shown in black in Figure 3. On the other hand, when the noise is treated as stochastic, then there are initial states from where the perturbed trajectory visits the target set B repeatedly. A trajectory with stochastic perturbation and the initial state I is also shown in the figure. The computation times are 417 sec for the over-approximation and 15 415 sec for the under-approximation.

Tool AMYTISS was also applied to Problem 3. It solves the problem in around 416 sec using CPU_1, 58 sec using CPU_1 and 4.7 sec using GPU_1.

5.7 Integrator-chain benchmark results

This year we have applied the integrator-chain benchmark on StocHy, AMYTISS, and SReachTools Kernel Module for the 2-D and 3-D versions.

Table 10: Comparison of tools on Integrator-chain benchmark in terms of computational time

N-dimensions	StocHy	AMYTISS	SReachTools Kernel Module
2	4.783	8.655	0.738
3	3225.80	66.71	0.754

Table 10 describes the results of the scalability comparison of StocHy, AMYTISS, and SReachTools Kernel Module. We can note an improvement of $\times 13.89$ for StocHy.

In addition, we have applied the SReachTools Kernel Module to solving the n-dimensional integrator chain problem for higher values of n. The SReachTools Kernel Module is more scalable than the pre-existing algorithms in SReachTools, and has the ability to analyze high-dimensional systems with up to 10,000-dimensions in less than 1 minute. The SReachTools Kernel Module successfully identified a safe initial state with reach probability ≈ 1 for $n \leq 10{,}000$. Because the algorithms in the SReachTools Kernel Module are sample-based, they do not rely upon a gridding-based approach, meaning the computational complexity scales exponentially with the sample size rather than the system dimension. The tool also has the ability to perform analysis of systems up to one million dimensions using a kernel speedup technique at the cost of higher approximation error.

5.8 7-Dimensional BMW 320i benchmark results

In this benchmark, we are interested in an autonomous operation of the vehicle to satisfy a reach-avoid property. In particular, the vehicle should park itself automatically in a parking lot located in the projected set $[-1.5, 0.0] \times [0.0, 1.5]$ within 32 time steps, while avoids hitting a barrier represented by the set $[-1.5, 0.0] \times [-0.5, 0.0]$. AMYTISS is the only tool presented in this report capable of handling this benchmark, and solved the problem in 825 sec. Since the dimension of the system is seven and relatively large for discretization-based techniques (the number of transitions in MDPs $|\hat{X} \times \hat{U}|$ is 3,937,500), the required memory for constructing the finite MDP would be very huge (and it is impossible in practice to construct such an abstraction due to the memory limitation). In order to handle this benchmark, we employ the *on-the-fly abstraction* technique as described in Section 2 to significantly reduce the required memory for constructing our finite MDP. We refer the interested reader to [54] for more details on the OFA technique.

5.9 Patrol robot benchmark results

In this benchmark, a robotic vehicle, modeled in Eq. (6), needs to autonomously patrol between two locations infinitely often, if the door between the two locations opens infinitely often (detailed specification is given in Eq. (7)). Mascot-SDS is the only tool presented in this report is able to perform a controller synthesis for this example. Mascot-SDS performs synthesis by computing a uniform finite-state abstraction of the continuous state systems, for which we fixed the dimension of the abstract states as $(0.1 \times 0.1 \times 0.1)$ and we sampled finitely many input points from the continuous input space with a discretization parameter 2.0. For this benchmark, we ran Mascot-SDS on a Macbook Pro (2015) with a 2,7 GHz Dual-Core Intel Core i5 processor and 16 GB RAM, and the total time for abstraction and synthesis was around 35 min.

5.10 Geometric Brownian motion benchmark results

Within ARCH2021 the geometric Brownian motion benchmark has been evaluated by SD-CPN&MC, SDCPN&IPS and AMYTISS. These results are given in Table 11 for the 10 levels for which Table 2 has given the analytical reach probabilities.

The second column in Table 11 shows the $\overline{\gamma}_{MC}$ results obtained through straightforward Monte Carlo (MC) simulation using 10000 runs with numerical integration time step $\Delta = 2 \times 10^{-3} s$. The results in Table 11 show that straightforward MC simulation based estimation of γ fails to work beyond $k = 4$.

Table 11: Results obtained for the geometric Brownian motion benchmark

	MC	IPS			AMYTISS
k	$\overline{\gamma}_{MC}$	$\hat{\gamma}$	ρ_s	$\hat{c}_{\hat{\gamma},NRMSE}$	upper bound
1	0.0957	9.78×10^{-2}	100%	10%	0.1401
2	0.0085	9.70×10^{-3}	100%	17%	0.0180
3	6.000×10^{-4}	9.58×10^{-4}	100%	28%	0.0080
4	1.000×10^{-4}	9.43×10-5	100%	44%	0.0148
5	0	9.18×10^{-6}	100%	66%	0.0253
6	0	8.98×10^{-7}	100%	94%	0.0409
7	0	8.65×10^{-8}	100%	131%	0.0635
8	0	8.23×10^{-9}	96%	179%	0.0954
9	0	7.68×10^{-10}	89%	235%	0.1396
10	0	7.17×10^{-11}	75%	301%	0.2010

The columns 3-5 in Table 11 show the results obtained through IPS, with $N_p = 1000$ particles. The applied IPS approach makes use of fixed assignment splitting (FAS) [63]. For the numerical integration of the SDE, use is made of a fixed numerical integration time step. This means that a stopping time of the process in hitting the k-th level may be somewhere halfway an integration time step. To keep the error low, a small time step of $\Delta = 4 \times 10^{-4} s$ is used. By repeating IPS $N_{IPS} = 1000$ times, we get estimates of the rate ρ_s of surviving IPS, and of a normalized root-mean-square error, $\hat{c}_{\hat{\gamma},NRMSE}$. The measures $\hat{\gamma}$, ρ_s and $\hat{c}_{\hat{\gamma},NRMSE}$ are defined as follows:

$$\hat{\gamma} = \frac{\sum_{i=1}^{N_{IPS}} \bar{\gamma}^i}{N_{IPS}} \tag{12}$$

$$\rho_S = \frac{\sum_{i=1}^{N_{IPS}} 1_{\bar{\gamma}^i > 0}}{N_{IPS}} \tag{13}$$

$$\hat{c}_{\hat{\gamma},NRMSE} = \frac{RMSE}{\gamma} \times 100\% \tag{14}$$

with $1_{\bar{\gamma}^i>0} = \begin{cases} 1, & \text{if } \bar{\gamma}^i > 0 \\ 0, & \text{if } \bar{\gamma}^i = 0 \end{cases}$ and

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N_{IPS}} (\bar{\gamma}^i - \gamma)^2}{N_{IPS}}} \tag{15}$$

where $\bar{\gamma}^i$ denotes the estimated reach probability for the i-th IPS run.

Application of AMYTISS to this case study. Since AMYTISS is developed for discrete-time models, we discretize the model in (8) with a sampling time $\hat{\tau}$ as

$$X(k+1) = X(k) + (\mu + \frac{\sigma^2}{2})\hat{\tau}X(k) + \sigma\sqrt{\hat{\tau}}X(k)w(k), \quad k \in \mathbb{N}, \tag{16}$$

where $w(\cdot) \sim \mathcal{N}(0, \mathbb{I}_n)$ is a sequence of independent random vectors with multivariate standard normal distributions (*i.e.,* mean zero and covariance matrix identity). This discrete-time model

can be used by any tool that computes the value of γ using analytical abstraction based methods, e.g., AMYTISS and StocHy.

We use to discretized model in (16) and try to approximate γ via two analytical approaches: (i) barrier certificate and (ii) reachability analysis via abstraction-based techniques and standard dynamic programming. We first use SOSTOOLS [69] together with a semi-definite programming (SDP) solver such as SeDuMi [84] to search for a polynomial barrier certificate [5, 67] with an initial set $[0.9, 1.1]$ and unsafe regions $[L_k, 1.1L_k]$. The probability of reaching the unsafe region in $T{=}1s$ via a potential barrier certificate can give us the rare event probability γ. However, we were not able to construct any *polynomial* barrier certificate. The main reason could be the shape of dynamics which is diverging *exponentially*, and as a result, polynomial barrier certificates cannot be found. In addition, it is not clear how one can implement an exponential barrier certificate in SOSTOOLS [69].

We now aim to do a change of variables to obtain a linear model for (16) and solve the reachability problem via standard dynamic programming using AMYTISS [54]. By taking operator log from both sides of (16), one has

$$\log X(k+1) = \log X(k) + \log\big[1 + \frac{3}{2}\hat{\tau} + \sqrt{\hat{\tau}}w(k)\big].$$

By defining

$$Y = \log X \to Y(k) = \log X(k), \quad \text{and } \log\big[1 + \frac{3}{2}\hat{\tau} + \sqrt{\hat{\tau}}w(k)\big] = \xi(k),$$

one has

$$Y(k+1) = Y(k) + \xi(k), \quad \text{with } Y(0) = 0.$$

Now we can directly compute the conditional probability density function of the system as

$$\begin{aligned}
\mathsf{T}(\bar{y} \mid y) &= \mathbb{P}\big\{\bar{Y} \le \bar{y} \mid y\big\} = \mathbb{P}\big\{y + \xi \le \bar{y} \mid y\big\} = \mathbb{P}\big\{\xi \le \bar{y} - y \mid y\big\} \\
&= \mathbb{P}\big\{\log\big[1 + \frac{3}{2}\hat{\tau} + \sqrt{\hat{\tau}}w\big] \le \bar{y} - y \mid y\big\} = \mathbb{P}\big\{1 + \frac{3}{2}\hat{\tau} + \sqrt{\hat{\tau}}w \le 10^{\bar{y}-y} \mid y\big\} \\
&= \mathbb{P}\big\{w \le \frac{10^{\bar{y}-y} - 1 - \frac{3}{2}\hat{\tau}}{\sqrt{\hat{\tau}}} \mid y\big\} = CDF\big(\frac{10^{\bar{y}-y} - 1 - \frac{3}{2}\hat{\tau}}{\sqrt{\hat{\tau}}}\big).
\end{aligned}$$

where the cumulative distribution function (CDF) of the standard normal distribution is used in the last line. Then the transition probabilities of the abstract system in AMYTISS can be computed as

$$\mathbb{P}\big\{a \le \bar{Y} \le b \mid y\big\} = \mathrm{CDF}\big(\frac{10^{b-y} - 1 - \frac{3}{2}\hat{\tau}}{\sqrt{\hat{\tau}}}\big) - \mathrm{CDF}\big(\frac{10^{a-y} - 1 - \frac{3}{2}\hat{\tau}}{\sqrt{\hat{\tau}}}\big), \tag{17}$$

By evaluating the CDF of normal distribution at two points (17) via AMYTISS and discretizing the state space, we can get an upper bound on the reach probability. This upper bound is reported in the right column of Table 11. Note that the provided bound is relatively good for the initial levels but become very pessimistic for other levels due to the increase in the size of the space that needs to be discretized. These upper bounds can be improved by increasing the precision of the abstraction (i.e., smaller partition sets of the state space), which requires extremely large memory and computational power. This experiment shows that current analytical methods based on abstractions are not able to handle rare event computations efficiently.

6 Conclusions

The achieved results this year are manifold. Three novel benchmarks with interesting properties and challenges were added to the collection of benchmarks for stochastic hybrid models. Thirteen tools were successfully evaluated on subsets of those collected benchmarks.

A recent survey of many of the discussed techniques can be found at [60], and provides the theoretical underpinnings for much of the computational results presented here.

6.1 Further tool development

6.1.1 Further development of AMYTISS

AMYTISS provides parallel algorithms that scale with respect to available compute resources, which enables handling more practical stochastic control problems. We will then use AMYTISS to design reach-avoid controllers for a real-world application. We plan to target the problem of real-time path-planning for a robot described by a simplified 4-dimensional version of the 7-dimensional BMW model. The robot should reach a predefined target while avoiding obstacles that might dynamically change. At the beginning of each control cycle (i.e., the sampling period) the robot should sense its state and the environment, solve the control problem using AMYTISS, generate a controller and deploys it. As the robot might be limited in compute capability, we may off-load the computation of the controller to HPC machines on the Cloud or ones that are available locally. In any of the cases, the network delay should be considered when planning for the real-time control. The implementation will be deployed on an actual robot. More specifically, the Amazon AWS DeepRacer car will be used as a target robot.

6.1.2 Further development of FAUST^2

FAUST^2 did not participate this year due to lack of resources. It will join the competition next year with further development on verification and synthesis for infinite horizon properties [91, 92] and stochastic systems with multiple players in a game theoretical setting [26, 62].

6.1.3 Further development of FIGARO

All the FIGARO workbench tools briefly described in section 2 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. [51]. This tool is used to experiment heuristics for choosing the states to keep in the partial construction of a large Markov chain [50].

6.1.4 Further development of HYPEG and hpnmg

For specific properties, as in the water sewage example, HYPEG can potentially speed-up the simulation significantly by stopping a simulation run when the property is already decided in contrast to simulating always until the maximum time. Furthermore, we are extending HYPEG to synthesize schedulers as in the new control synthesis version of the heated tank. The tool hpnmg will be extended towards more general stochastic hybrid systems, such that also discrete nondeterminism and nondeterminism through time delays can be resolved to optimize reachability probabilities. Furthermore, we will investigate the use of different state-set representations.

6.1.5 Further development of Mascot-SDS

Despite being powerful in approximating the solution of the rich class of ω-regular (infinite-horizon) specifications, the latest release (version 1.1) of Mascot-SDS can only approximate the initial states from which the specification can be satisfied with probability 1. Future extension of this tool will include the quantitative aspect of the problem as well (i.e. computing the optimal probability of satisfying the specification for any initial state). Moreover, the current version is not scalable to very large dimensional systems due to the curse of dimensionality introduced by the discrete abstraction. Future release will use intensive parallelization of the abstraction and the synthesis procedure to mitigate the scalability issue.

6.1.6 Further development of modes and prohver in the Modest Toolset

modes is currently being extended with learning-based methods to tackle the new control synthesis version of the heated tank. We would also like to more thoroughly support limited-information control policies (such as distributed and non-prophetic schedulers), and export them in useful formats (e.g. as decision trees). For prohver, which was not applied this year, the main challenge remains to replace the current usage of PHAVer as a backend by a maintained, up-to-date tool.

6.1.7 Further development of ProbReach

The main aim for next year is to fix the computation of rigorous enclosures, *i.e.*, upper and lower bounds for probabilities. Currently, the lower bound computation is affected by a problem with the generation of the logical formulae that are passed to the SMT solver (these are non-trivial, exists-forall quantified formulae). We also plan to introduce a "upper bounding" modality for ProbReach to compute only upper bounds for probabilities. This would avoid the more complex computations needed for calculating lower bounds, thereby speeding up ProbReach. This option could be especially useful when calculating rare-event probabilities.

6.1.8 Further development of PYCATSHOO

PyCATSHOO is used at EDF by several safety assessment tools for hydraulic and nuclear power plants and electric networks. However, its improvement and extension of its functionalities are still in progress. Our current main project is carried out in the framework of thesis work. It aims at developing robust acceleration algorithms for Monte Carlo simulation and efficient sensitivity indices for rare event assessment in PDMP. On the other hand, we are currently integrating into PyCATSHOO efficient capabilities for modelling and solving Markov decision processes. Finally, we are developing a hosting platform for the tools developed with PyCATSHOO. This platform will provide to these tools the functionalities of a graphical interface for modelling, post-processing, and other productivity features.

6.1.9 Further development of SReachTools

We will continue to develop SReachTools to provide abstraction-free, convex-optimization-based methods for verification and stochastic reachability analysis of discrete-time stochastic switched systems and Markov jump affine systems. Notably, the addition of the SReachTools kernel module opens the door for data-driven stochastic reachability analysis of a wider variety of systems than those previously considered, such as systems with nonlinear dynamics and arbitrary stochastic disturbances. We plan to extend recent results for efficient, abstraction-free forward

stochastic reachability analysis using Fourier transforms [98] and interpolation of stochastic reach sets [99]. In addition, we will continue to develop the SReachTools Kernel Module to provide statistical-learning-based algorithms for stochastic reachability analysis of discrete-time stochastic systems and will incorporate recent results based in functional analysis to enable controller synthesis and computation of stochastic reach sets [87, 88].

6.1.10 Further development of StocHy

In this year's edition, StocHy's means of computation of the underlying transition probabilities and hence the abstraction process has been parallelised. This has resulted in a significant speed up in the abstraction generation with an average speed up of $\times 13$. Further work is however required to reduce the number of states generated for the underlying abstraction and to simplify the construction of the models with non-linear dynamics.

6.1.11 Further development of SysCore

SysCore can be used to compute finite and reduced order abstractions. Within SysCore, the accuracy of these abstractions can be quantified with epsilon, delta simulation relations. Furthermore, these abstractions can be used for control synthesis with robust guarantees that can be leveraged over the original model. Thereby, SysCore enables control refinement with guarantees based on both reduced order and finite state stochastic models.

SysCore will be extended to inter alia nonlinear stochastic model classes with different stochastic distributions. Additionally, moving beyond the current abstraction to finite and reduced order models, abstractions to deterministic models will also be added together with more advanced multi-layered computations.

6.1.12 Benchmarks that pose novel challenges

In this edition of the competition, we also introduced two novel benchmarks for future evaluation of tools. One is an adaptation of the stochastic Van der Pol oscillator to a benchmark version that challenges rare-event estimation tools. The other is an adaptation of the Heated Tank to a benchmark version that challenges tools that not only estimates reach probability but also enable the synthesis of control actions or parameters for optimizing such probabilities. These two novel benchmarks have been prepared for future tools evaluations. We expect that they will motivate further development of the tools and improve the capabilities of the techniques currently available. We also welcome any new tool or novel approach that implement data-driven methods for verification and controller synthesis of stochastic systems against complex temporal properties.

References

[1] 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.

[2] 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.

[3] 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.

[4] Alessandro Abate, Maria Prandini, John Lygeros, and Shankar Sastry. Probabilistic reachability and safety for controlled discrete time stochastic hybrid systems. *Automatica*, 44(11):2724–2734, 2008.

[5] M. Anand, A. Lavaei, and M. Zamani. Compositional construction of control barrier certificates for large-scale interconnected stochastic systems. *IFAC-PapersOnLine*, 53(2):1862–1867, 2020.

[6] 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.

[7] H.A.P. Blom, H. Ma, and G.J. Bakker. Interacting particle system-based estimation of reach probability for a generalized stochastic hybrid system. *IFAC-PapersOnLine*, 51(16):79–84, 2018.

[8] Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. MoDeST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.*, 32(10):812–830, 2006.

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

[14] Carlos E. Budde, Pedro R. D'Argenio, and Arnd Hartmanns. Automated compositional importance splitting. *Sci. Comput. Program.*, 174:90–108, 2019.

[15] 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.

[16] Carlos E Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. Jani: Quantitative model and tool interaction. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 151–168. Springer, 2017.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

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

[26] J. Ding, M. Kamgarpour, S. Summers, A. Abate, J. Lygeros, and C.J. Tomlin. A stochastic games framework for verification and control of discrete-time stochastic hybrid systems. *Automatica*, 49(9):2665–2674, 2013.

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

[28] 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.

[29] 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.

[30] 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.

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

[32] Hamed Ghasemieh, Anne Remke, and Boudewijn Haverkort. Analysis of a sewage treatment facility using hybrid Petri nets. In *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*. ICST, 2014.

[33] Hamed Ghasemieh, Anne Remke, and Boudewijn R. Haverkort. Survivability Evaluation of Fluid Critical Infrastructures Using Hybrid Petri Nets. In *19th IEEE Pacific Rim International Symposium on Dependable Computing, PRDC 2013*, pages 152–161. IEEE, 2013.

[34] Joseph D. Gleason, Abraham P. Vinod, and Meeko M. K. Oishi. Underapproximation of reach-avoid sets for discrete-time stochastic systems via lagrangian methods. In *IEEE Conference on Decision and Control (CDC)*, pages 4283–4290, Dec 2017.

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

[36] Sofie Haesaert, Nathalie Cauchi, and Alessandro Abate. Certified policy synthesis for general Markov decision processes: An application in building automation systems. *Performance Evaluation*, 117:75–103, 2017.

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

[38] 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.

[39] Sofie Haesaert, Sadegh 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.

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

[41] 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.

[42] Kyle Hsu, Rupak Majumdar, Kaushik Mallik, and Anne-Kathrin Schmuck. Multi-layered abstraction-based controller synthesis for continuous-time systems. In *Proceedings of the 21st International Conference on Hybrid Systems: Computation and Control (Part of CPS Week)*, HSCC '18, page 120–129, New York, NY, USA, 2018. Association for Computing Machinery.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] S. Karlin and H.M. Taylor. Chapter 7 - brownian motion. In S. Karlin and H. M. Taylor, editors, *A First Course in Stochastic Processes*, pages 340–391. Academic Press, 1975.

[49] M. Khaled and M. Zamani. `pFaces`: An acceleration ecosystem for symbolic control. In *Proceedings of the 22nd ACM International Conference on Hybrid Systems: Computation and Control*, HSCC '19, New York, NY, USA, 2019. ACM.

[50] S. Khan, M. Volk, J.P. Katoen, and M. Bouissou. Scalable reliability analysis by lazy verification. In *Proceedings of 13th NASA Formal Methods Symposium (NFM 2021)*, 2021.

[51] 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.

[52] J. Krystul. *Modelling of stochastic hybrid systems with applications to accident risk assessment.* PhD thesis, University of Twente, 2006.

[53] L. Laurenti, M. Lahijanian, A. Abate, L. Cardelli, and M. Kwiatkowska. Formal and efficient control synthesis for continuous-time stochastic processes. *IEEE Transactions on Automatic Control*, 66(1):17–32, 2021.

[54] 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.

[55] A. Lavaei, S. Soudjani, and M. Zamani. From dissipativity theory to compositional construction of finite Markov decision processes. In *Proceedings of the 21st ACM International Conference on Hybrid Systems: Computation and Control*, pages 21–30, 2018.

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

[57] A. Lavaei, S. Soudjani, and M. Zamani. Compositional abstraction-based synthesis of general MDPs via approximate probabilistic relations. *Nonlinear Analysis: Hybrid Systems*, 2020.

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

[59] A. Lavaei, S. Soudjani, and M. Zamani. Compositional (in)finite abstractions for large-scale interconnected stochastic systems. *IEEE Transactions on Automatic Control*, 65(12):5280–5295, 2020.

[60] Abolfazl Lavaei, Sadegh Soudjani, Alessandro Abate, and Majid Zamani. Automated verification and synthesis of stochastic hybrid systems: A survey, 2021.

[61] G. Peter Lepage. A new algorithm for adaptive multidimensional integration. *Journal of Computational Physics*, 27(2):192–203, 1978.

[62] K. Lesser and A. Abate. Multi-objective optimal control with safety as a priority. *IEEE Transactions on Control Systems Technology*, 26(3):1015–1027, 2018.

[63] H. Ma and H.A.P. Blom. Random assignment vs. fixed assignment in multilevel importance splitting for estimating stochastic reach probabilities, submitted 1 june 2021.

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

[65] 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.

[66] 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.

[67] A. Nejati, S. Soudjani, and M. Zamani. Compositional construction of control barrier functions for networks of continuous-time stochastic systems. *IFAC-PapersOnLine*, 53(2):1856–1861, 2020.

[68] 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.

[69] A. Papachristodoulou, J. Anderson, G. Valmorbida, S. Prajna, P. Seiler, and P. Parrilo. SOSTOOLS version 3.00 sum of squares optimization toolbox for MATLAB. *arXiv:1310.4716*, 2013.

[70] 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.

[71] 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.

[72] 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.

[73] Hossein Sartipizadeh, Abraham P. Vinod, Behçet Açikmese, and Meeko Oishi. Voronoi partition-based scenario reduction for fast sampling-based stochastic reachability computation of LTI systems. In *Proceedings of the American Control Conference*, pages 37–44, 2019.

[74] 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.

[75] S.Haesaert, P. Nilsson, Cristian-Ioan Vasile, Rohan Thakker, Ali akbar Agha-mohammadi, Aaron D. Ames, and Richard M. Murray. Temporal logic control of POMDPs via label-based stochastic simulation relations. In *IFAC Conference on Analysis and Design of Hybrid Systems*, 2018.

[76] Fedor Shmarov and Paolo Zuliani. ProbReach: Verified probabilistic δ-reachability for stochastic hybrid systems. In *HSCC*, pages 134–139. ACM, 2015.

[77] Fedor Shmarov and Paolo Zuliani. Probabilistic hybrid systems verification via SMT and Monte Carlo techniques. In *HVC*, volume 10028 of *LNCS*, pages 152–168, 2016.

[78] Fedor Shmarov and Paolo Zuliani. SMT-based reasoning for uncertain hybrid domains. In *AAAI-16 Workhop on Planning for Hybrid Systems*, pages 624–630, 2016.

[79] 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.

[80] 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.

[81] 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.

[82] Sadegh Soudjani and Alessandro Abate. Probabilistic invariance of mixed deterministic-stochastic dynamical systems. In *ACM Proceedings of the 15th International Conference on Hybrid Systems: Computation and Control*, pages 207–216, Beijing, PRC, April 2012.

[83] Sadegh Soudjani, Alessandro Abate, and Rupak Majumdar. Dynamic Bayesian networks for formal verification of structured stochastic processes. *Acta Informatica*, 54(2):217–242, Mar 2017.

[84] J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization methods and software*, 11(1-4):625–653, 1999.

[85] Sean Summers and John Lygeros. Verification of discrete time stochastic hybrid systems: A stochastic reach-avoid decision problem. *Automatica*, 46(12):1951–1961, 2010.

[86] Adam J. Thorpe and Meeko M. K. Oishi. Model-free stochastic reachability using kernel distribution embeddings. *IEEE Control Systems Letters*, 4(2):512–517, 2020.

[87] Adam J Thorpe and Meeko MK Oishi. Stochastic optimal control via hilbert space embeddings of distributions. *arXiv preprint arXiv:2103.12759*, 2021.

[88] Adam J Thorpe, Kendric R Ortiz, and Meeko MK Oishi. Learning approximate forward reachable sets using separating kernels. In *Learning for Dynamics and Control*, pages 201–212. PMLR, 2021.

[89] Adam J Thorpe, Kendric R Ortiz, and Meeko MK Oishi. SReachTools Kernel Module: Data-driven stochastic reachability using Hilbert space embeddings of distributions. *arXiv preprint arXiv:2011.10610*, 2021.

[90] Adam J Thorpe, Vignesh Sivaramakrishnan, and Meeko MK Oishi. Approximate stochastic reachability for high dimensional systems. In *Proceedings of the American Control Conference*, 2021.

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

[92] 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.

[93] H.C. Tuckwell and F.Y. Wan. First-passage time of markov process to moving barriers. *Journal of applied probability*, 21(4):695–709, 1984.

[94] 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.

[95] Abraham P. Vinod, Joseph D. Gleason, and Meeko M. K. Oishi. SReachTools: A MATLAB Stochastic Reachability Toolbox. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 33 – 38, Montreal, Canada, April 16–18 2019.

[96] Abraham P. Vinod, Baisravan HomChaudhuri, and Meeko M. K. Oishi. Forward stochastic reachability analysis for uncontrolled linear systems using Fourier transforms. In *Proceedings of the 20th International Conference on Hybrid Systems: Computation and Control, HSCC*, pages 35–44, April, 2017.

[97] Abraham P. Vinod and Meeko M. K. Oishi. Scalable underapproximation for the stochastic reach-avoid problem for high-dimensional LTI systems using Fourier transforms. *IEEE Control Systems Letters*, 1(2):316–321, Oct 2017.

[98] Abraham P. Vinod and Meeko M. K. Oishi. Probabilistic occupancy via forward stochastic reachability for Markov jump affine systems. *IEEE Transactions on Automatic Control (accepted)*, 2018. Available online: https://arxiv.org/abs/1803.07180.

[99] Abraham P. Vinod and Meeko M. K. Oishi. Stochastic reachability of a target tube: Theory and computation. *Automatica (in review)*, 2018. Available online: https://arxiv.org/abs/1810.05217.

[100] Abraham P. Vinod and Meeko M. K. Oishi. Scalable underapproximative verification of stochastic LTI systems using convexity and compactness. In *Proceedings of the International Conference on Hybrid Systems: Computation and Control*, pages 1–10. ACM, April, 2018.

[101] H. Zhang, F. Dufour, Y. Dutuit, and K. Gonzalez. Piecewise deterministic markov processes and dynamic reliability. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 222(4):545–551, 2008.