Constraint-Solving Techniques for the Analysis of Stochastic Hybrid Systems

Martin Fränzle, Yang Gao and Sebastian Gerwinn

Abstract The ProCoS project has been seminal in widening the perspective on verification of computer-based systems to a coverage of the detailed interaction and feedback dynamics between the embedded system and its environment. We have since then seen a steady increase both in expressiveness of the "hybrid" modeling paradigms adopting such an integrated perspective and in the power of automatic reasoning techniques addressing relevant fragments of logic and arithmetic. In this chapter we review definitions of stochastic hybrid automata and of parametric stochastic hybrid automata, both of which unify the hybrid view on system dynamics with stochastic modeling as pertinent to reliability evaluation, and we elaborate on automatic verification and synthesis methods based on arithmetic constraint solving. The procedures are able to solve step-bounded stochastic reachability problems and multi-objective parameter synthesis problems, respectively.

1 Introduction

An increasing number of the technical artifacts shaping our ambience are relying on often invisible embedded computer systems, rendering embedded computers the most common form of computing devices today. The vast majority — 98% according

This research has partially been funded by the German Research Foundation through the Collaborative Research Action SFB-TR 14 "Automatic Verification and Analysis of Complex Systems" (AVACS, www.avacs.org) and the Research Training Group DFG-GRK 1765: "System Correctness under Adverse Conditions" (SCARE, `scare.uni-oldenburg.de`).

M. Fränzle (✉) · Y. Gao
Department of Computing Science, Carl von Ossietzky Universität Oldenburg, Oldenburg, Germany
e-mail: fraenzle@informatik.uni-oldenburg.de

Y. Gao
e-mail: yang.gao@informatik.uni-oldenburg.de

S. Gerwinn
OFFIS Institute for Information Technology, Oldenburg, Germany
e-mail: sebastian.gerwinn@offis.de

M.G. Hinchey et al. (eds.), *Provably Correct Systems*, NASA Monographs in Systems and Software Engineering, DOI 10.1007/978-3-319-48628-4_2

to www.artemis-ju.eu — of all processing elements manufactured goes to embedded applications, where they monitor and control all kinds of physical processes. Such interactions of the virtual with the physical world range from traditional control applications, like controlling an automotive powertrain, over computer-controlled active safety systems, like the anti-locking brake, the electronic stability program, or recently pedestrian detection integrated with emergency braking capabilities, to the vision of cyber-physical networks bringing even remote physical processes into our sphere of control.

Even to the general public, it has become evident that such immersion of computing elements into physical environments renders their functionality critical in many respects: critical to the function of the overall product, where a malfunction or undesired interaction of the embedded system may render the whole product partially or totally dysfunctional; critical to the performance of the overall product, where embedded control may influence power consumption, environmental impact, and many more performance dimensions; finally safety-critical, as causal chains mediated by the physical environment may propagate software faults and thus endanger life and property. A direct consequence is that correctness (in a broad sense) of embedded systems most naturally is defined in terms of the possible physical impact of its interaction with the environment. This insight marked a paradigm shift in the attitude to software correctness at the times when the ProCoS project was conceived a good quarter of a century ago. Rather than saying that correct software ought to infallibly implement some abstract algorithm or establish correctness properties in terms of conditions on intrinsic program variables (and other notions intrinsic to the algorithm, like termination), correctness became defined in terms of observables of external physical processes underlying an independent dynamics extrinsic to and only partially controllable by the software.

Within the ProCoS project, this issue was taken up by Zhou Chaochen and Anders Ravn, who first together with Tony Hoare defined the Duration Calculus [9] and later together with Hans Rischel and Michael R. Hansen extended it to cover hybrid discrete-continuous phenomena [10, 34]. While the former took the perspective of durational metric-time properties at the interface between embedded system and environment, thus not really covering environmental dynamics, the extension to Extended Duration Calculus in [10] permitted integration about environmental variables and thus formulation of integral equations, as equivalent to initial-value problems of ordinary differential equations. It thus is an example of a formal model permitting to model and analyze the tight interaction, and hence feedback dynamics, of the discrete switching behavior of embedded systems and the continuous dynamics of the physical environment as well as of continuous control components embedded into it. A related model also confining the description of environmental dynamics to ordinary differential equations and also adopting a qualitative view of the embedded system as a potentially demonically non-deterministic discrete computational device is Hybrid Automata [1]. Such models support *qualitative* behavioral verification in the sense of showing that a system behaving according to its nominal dynamics would *never* be able to engage in an undesired behavior, which, however, is a goal unlikely to be achieved in practice. First, designing systems to that level of

correctness may be prohibitively expensive or it may be impractical due to necessary inclusion of components lacking tangible models of nominal behavior, like, e.g., computer vision and image classification algorithms. Second, even when possible in principle, the systems verified to that level will necessarily eventually deviate from their nominal behavior due to, e.g., component failures. Reflecting the consequential need for *quantitative* verification, stochastic variants of hybrid-system models have been suggested, like Probabilistic Hybrid Automata (PHA) [35] or Stochastic Hybrid Automata (SHA) [25]. In such extensions, either discrete actions can feature probabilistic branching or continuous evolution can evolve stochastic along, e.g., stochastic differential equations, or both.

The resulting models are inherently hard to analyze, as they combine various sources of undecidability, like state reachability in even the simplest classes of hybrid automata and the fragments of arithmetic induced by ordinary differential equations, with the necessity of reasoning about probability distributions over complexly shaped and sometimes not even first-order definable carriers, like the reachable states. It is thus obvious that exact automatic analysis methods for properties of interest, like the probability of reaching undesirable states, are impossible to attain. Nevertheless, safe approximations can be computed effectively, and often prove to be of sufficient accuracy to answer relevant questions with scrutiny, like certifying that the probability of reaching undesirable states remains below a given quantitative safety target. The pertinent techniques do either rely on state-space discretization by safe abstraction, e.g. Hahn et al.'s approach [43], or on constraint solving for stochastic logic involving arithmetic [15, 19], or on massive simulation paired with statistical hypothesis testing, beginning with Younes' seminal work [42]. We will in the remainder of this chapter report on our contributions to the constraint-based approach, thereby building on a series of results obtained over the past decade.

Structure of the Chapter

In the next section, we provide an introduction to a class of stochastic hybrid automata featuring stochasticity —paired with non-determinism or parametricity— in their transitions. In Sect. 3, we move on to the depth-bounded safety analysis of such stochastic hybrid automata. The underlying technology is an extension of satisfiability-modulo-theory solving (SMT, [4]) to Stochastic Satisfiability-Modulo-Theory (SSMT) akin to the extension of Propositional Satisfiability (SAT) to Stochastic Propositional Satisfiability (SSAT) suggested by Papadimitriou and Majercik [28, 33]. Section 4, finally, turns to the problem of multi-objective parameter synthesis in parametric variants of stochastic hybrid automata, which we solve by a machine-learning style integration of simulation and arithmetic constraint solving.

2 Stochastic Hybrid Transition Systems

The model of hybrid automata [1, 2, 22, 26] has been suggested as a formal set-up for analyzing the the interaction of discrete and continuous processes in hybrid-state dynamical systems. They combine pertinent formalism for describing discrete,

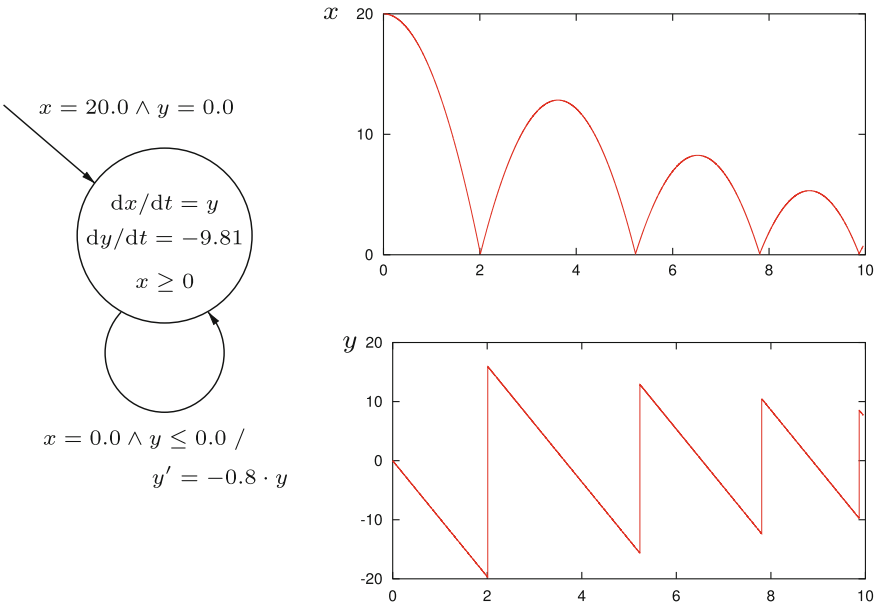
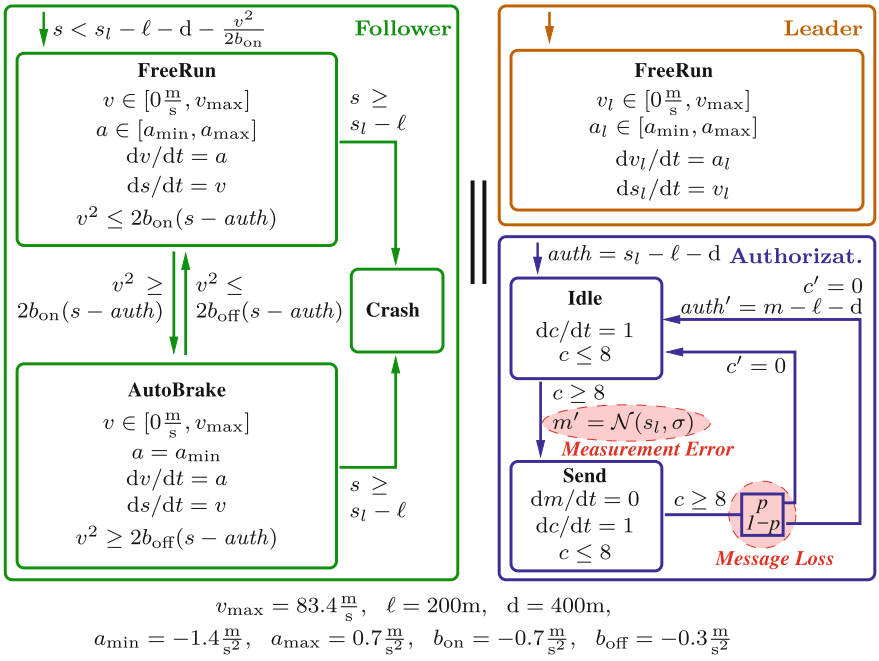


Fig. 1 A simple hybrid automaton (*left*) and a trajectory thereof (*right*)

computational and continuous, mostly physical or control-oriented dynamical processes by extending finite automata with a vector of continuous variables and "decorating" them with ordinary differential equations in each location and assignments to these extra variables upon transitions. A simple hybrid automaton and its associated dynamic behavior, which is a piece-wise continuous trajectory, are depicted in Fig. 1.

While this model permits the analysis of deterministic as well as uncertain hybrid-state systems, as the latter can be modeled by various forms of non-determinism in the automata, like uncontrolled inputs, non-deterministic transition selection, or parameter ranges in the differential (in-)equations, it is confined to *qualitative* behavioral verification in the sense of showing that a system behaving according to its nominal dynamics would never be able to engage in an undesired behavior. This ideal, however, is hardly achieved in practice, as systems strictly adhering to their nominal behavior would either be prohibitively expensive or even infeasible to design. Qualitative verification consequently is indicative of the nominal behavior only, yet does not cover the full set of expected behaviors of the system under design, which includes (traditionally rare, but with the advent of trained classifiers in, e.g., computer vision systems for automated driving increasingly frequent) deviations from nominal behavior also.

As the expected low to moderate frequency of deviations from nominal behavior would not justify the same demonic view of uncertainties in system dynamics as adopted in qualitative verification, namely that every single abnormal behavior that



$$v_{\max} = 83.4\tfrac{\mathrm{m}}{\mathrm{s}}, \quad \ell = 200\mathrm{m}, \quad \mathrm{d} = 400\mathrm{m},$$
$$a_{\min} = -1.4\tfrac{\mathrm{m}}{\mathrm{s}^2}, \quad a_{\max} = 0.7\tfrac{\mathrm{m}}{\mathrm{s}^2}, \quad b_{\mathrm{on}} = -0.7\tfrac{\mathrm{m}}{\mathrm{s}^2}, \quad b_{\mathrm{off}} = -0.3\tfrac{\mathrm{m}}{\mathrm{s}^2}$$

Fig. 2 Model of moving-block train control in ETCS level 3 including relevant random disturbances (encircled areas) in the form of measurement error in positional information and possible message loss, after [17]

might be possible would render the system design incorrect, *quantitative* counterparts to qualitative hybrid models and verification methods have been developed. The corresponding *probabilistic* or *stochastic* models provide more concise quantitative information about the uncertainties involved in terms of probabilities. To incorporate this kind of information, both the underlying models and the corresponding analysis techniques have to be adapted. Verifying reachability and safety properties within this extended setting then corresponds to obtaining statements about the probability of these properties to be satisfied.

To illustrate the challenges arising in incorporating the information about random disturbances into the model of hybrid automata, we show a model model mimicking distance control at level 3 of the European train control systems ETCS in Fig. 2. The idea of the control system is to switch to an automatic braking mode "AutoBrake" initiating a controlled emergency deceleration whenever the necessary deceleration rate for coming to a standstill at a safety distance (400 m) behind the preceding train exceeds a threshold value ($-0.7\ \frac{\mathrm{m}^2}{\mathrm{s}}$). The function of this control system, however, is impeded by measurement noise in determining train positions and the risk of message loss between trains, as positions are determined locally in a train and announced via train-to-train communication. The extended hybrid model depicted in Fig. 2 incorporates these stochastic disturbances. More specifically, the perturbed measurement

of the position of the leading train is characterized by a normal distribution $\mathcal{N}(s_l, \sigma)$ centered around the true position s_l and the measurement process itself is modeled by copying this skewed image of the physical entity s_l into its real-time image m. This is in contrast to a typical nominal model, where the controller may be modeled as having direct access to the physical entities. Additionally, unreliable communication is also considered, i.e., the communication of resultant movement authorities is allowed to fail with probability 0.1. The resulting model is called a *Stochastic Hybrid Automaton (SHA)* [25, 35]. Note that the automaton in Fig. 2 features both non-determinism and stochasticity in its transitions, with the former being interpreted demonically.

For such models, we are interested in solving two problems, which will be the subjects of Sects. 3 and 4:

1. Given a stochastic hybrid automaton A and a set of undesirable states G in the state set of A, determine whether the probability of reaching G stays below a given safety target ε. Given that non-determinism is interpreted demonically, the probability of reaching G thereby has to be determined w.r.t. a most malicious adversary resolving the non-deterministic choices.
2. Given a stochastic hybrid automaton featuring parameters in its probability distributions, determine whether there is a parameter instance satisfying a design objective in terms of expectations on some cost and/or reward variables in the hybrid system.

Formally, SHA are infinite-state Markov Decision Processes (MDP), where the infinite-state behavior is induced by the hybrid discrete-continuous state dynamics, while the MDP property arises from the interplay of stochastic and non-deterministic choices. A stochastic hybrid system (in its continuous-time variant) thus interleaves

1. *continuous flows* arising while residing in a discrete location and being governed by the differential (in-)equation and the invariant condition assigned to the location with
2. *immediate transitions* featuring a guard condition on the continuous variables, a deterministic or non-deterministic state update w.r.t. both some continuous variables and possibly the discrete successor location, and potentially a series of randomized updates to continuous variables are the successor location.

As a suitable semantic basis for the automatic analysis of hybrid automata featuring stochastic behavior, we can consequently base our investigations on a more abstract definition of hybrid-state transition systems featuring stochastic behavior, a form of infinite-state Markov Decision Process (MDP). In full generality, such a (parametric) hybrid stochastic transition system comprises the following:

1. A finite set $D = \{d_1, \ldots .d_m\}$ of discrete variables. Discrete variables range over $\mathbb{Z}$.[1]

[1]The reader might expect to rather see finite sub-ranges of $\mathbb{Z}$ or other finite sets as domains. To avoid cluttering the notation, we abstained from this. It should be noted that this does not induce a loss of generality, as not all of $\mathbb{Z}$ need to be dynamically reachable.

2. A finite set $C = \{x_1, \ldots, x_n\}$ of continuous variables. Continuous variables range over $\mathbb{R}$.[2]
3. A finite (and possibly empty) set $P = \{p_1, \ldots, p_k\}$ of parameters with associated range $\theta \subseteq \mathbb{R}^k$.
4. An initial state $i \in \Sigma$, where $\Sigma = \mathbb{Z}^m \times \mathbb{R}^n$ is the state set of the transition system.
5. A finite set $T = \{t_1, \ldots, t_l\}$ of stochastic transitions. Each such transition comprises a non-deterministic guarded assignment, expressed as a pre-post relation in $\Sigma \times \Sigma$, followed by a finite (possibly empty) sequence of stochastic assignments to individual variables, which are executed in sequence and may depend on the preceding ones and on the parameters.

Traditional stochastic hybrid automata diagrams, as depicted in Figs. 2 and 6, can easily be interpreted as instances of this model by interpreting both their continuous flows and immediate transitions as stochastic transitions. To this end, please note that stochastic transitions need not contain a proper stochastic part, but may also be just non-deterministic or even deterministic.

In order to achieve a uniform treatment of discrete and continuous stochastic assignments, we equip $\mathbb{R}$ with the Lebesgue measure and $\mathbb{Z}$ with cardinality of its subsets as a measure. Given this convention, we can uniformly write $\int_a^b p(x)\mathrm{d}x$ for determining the probability mass assigned by a density (or, in the discrete case, a distribution) p to the interval $[a, b]$, as the measure is understood. Note that in the discrete case, $\int_a^b p(x)\mathrm{d}x = \sum_{x=a}^{b} p(x)$ due to the particular choice of the measure for $\mathbb{Z}$. This permits us to uniformly treat densities over the continuum and distributions over discrete carriers as densities. A density over domain X, where X is either $\mathbb{R}$ or $\mathbb{Z}$, is a measurable function $\delta : X \to \mathbb{R}_{\geq 0}$ with $\int_{-\infty}^{\infty} \delta(x)\mathrm{d}x = 1$. We denote by P_X the set of all densities over X. A stochastic assignment for a variable $v \in D \cup C$ with its associated domain $V \in \{\mathbb{Z}, \mathbb{R}\}$ is a mapping $sa_v : \theta \to \Sigma \to P_V$. It assigns to each parameter instance and each state a density of the successor values for v.

We are ultimately interested in determining the probability of reaching a certain set $G \subset \Sigma$ of goal states or the expectation of a function $f : \sigma \to \mathbb{R}$. As the former is a special case of the latter, using the characteristic function χ_G[3] as reward, it suffices to define expectations.

Given that a non-deterministic assignment simply is a relation between pre- and post-states, i.e., a subset of $\Sigma \times \Sigma$ defining both the transition guard (due to possible partiality of the relation) and the (possibly non-deterministic) update to all the variables, depth-bounded expectations in the infinite-state MDP mediated by the stochastic hybrid transition system can now be defined inductively by means of a Bellmann backward induction [5] as follows: The (best-case) expectation $\mathcal{E}_f^k(\sigma, \theta)$ of reward f over k steps of the transition system under parameterization θ and starting from state $\sigma \in \Sigma$ is

[2] As for discrete variables, this does not exclude the possibility that only a bounded sub-range may dynamically be reachable.

[3] Defined as $\chi_G(\sigma) = \begin{cases} 1 & \text{if } \sigma \in G, \\ 0 & \text{if } \sigma \notin G. \end{cases}$

$$\mathcal{E}_f^0(\sigma,\theta) = f(\sigma)\ ,$$
$$\mathcal{E}_f^{k+1}(\sigma,\theta) = \max_{t=\langle na, sa_1,\ldots,sa_n\rangle\in T} \max_{\sigma_1\in\Sigma \text{ such that } (\sigma,\sigma_1)\in na} \int\ldots\int sa_1^\theta(\sigma_1)(\sigma_2)\cdots sa_n^\theta(\sigma_n,)(\sigma_{n+1})\mathcal{E}_f^k(\sigma_{n+1},\theta)\mathrm{d}\sigma_1\ldots\mathrm{d}\sigma_{n+1}\ .$$

Here, na denote the non-deterministic assignment and sa_i^θ denotes the effect of a stochastic assignment, resp., in a transition $t = \langle na, sa_1, \ldots, sa_n\rangle$. The effect sa^θ of a stochastic assignment sa to variable v is

$$sa^\theta(\sigma)(x) = \begin{cases} \sigma(x) & \text{if } x \neq v, \\ sa(\theta)(\sigma)(v) & \text{if } x = v. \end{cases}$$

Taken together, the non-deterministic transition selection as well as the non-deterministic assignment thus implements an oracle maximizing rewards, while the stochastic assignments just implement their stochastic transition kernels.

3 Bounded Reachability Checking for Stochastic Hybrid Automata

In order to analyze Stochastic Hybrid Automata (SHA) models, like the one depicted in Fig. 2, we need techniques being able to analyze their intrinsic combination of stochastic dynamics and infinite-state behavior. Formally, SHA are infinite-state Markov Decision Processes (MDP), where the infinite-state behavior is induced by the hybrid discrete-continuous state dynamics, while the MDP property arises from the interplay of stochastic and non-deterministic choices (e.g., concerning a in Fig. 2). As verification tools for finite-state MDP are readily available, a particular technique is to use abstraction for obtaining a safe finite-state overapproximation, subsequently verifying the properties of interest on the abstraction, as pursued e.g. in [17]. A more direct approach along the lines of bounded model checking (BMC) [6, 21] in its variant for hybrid automata [3, 12, 23] is to encode the stochastic behavior within the constraint formula. This requires more expressive constraint logic then the satisfiability-modulo-theory calculi used in the case of qualitative verification [3, 12, 23, among others], which have been pioneered by Fränzle, Hermanns, and Teige under the name *Stochastic Satisfiability Modulo Theory (SSMT)* [15].

3.1 Stochastic Satisfiability Modulo Theory

The idea of modeling uncertainty in satisfiability problems was first proposed within the framework of propositional satisfiability (SAT) by Papadimitriou, yielding

Stochastic SAT (SSAT) [33], a logic featuring both existential quantifiers and randomized quantifiers allowing to express $1\frac{1}{2}$ player games (one strategic, one randomized player). This work has been lifted to Satisfiability Modulo Theories (SMT) by Fränzle, Teige et al. [15, 38], providing a logic called Stochastic Satisfiability Modulo Theory (SSMT) permitting symbolical reasoning about bounded reachability problems of probabilistic hybrid automata (PHA). Instead of being true or false, an SSAT or SSMT formula Φ has a probability as semantics. This quantitative semantics reflects the probability of satisfaction of Φ under optimal resolution of the non-random quantifiers. SSAT and SSMT permit concise description of diverse problems combining reasoning under uncertainty with data dependencies. Applications range from AI planning [27, 29, 30] to analysis of PHA [15]. A major limitation of the SSMT-solving approach pioneered by Teige [37] is that all quantifiers (except for implicit innermost existential quantification of all otherwise unbound variables) are confined to range over finite domains. As this implies that the carriers of probability distributions have to be finite, a large number of phenomena cannot be expressed within that SSMT framework, such as continuous noise or measurement error in hybrid systems. To overcome this limitation, our recent work [19] relaxes the constraints on the domains of randomized variables, now also admitting quantification over continuous domains and continuous probability distributions in SSMT solving.

The approach is based on a combination of the iSAT arithmetic constraint solver [13] with branch-and-prune rules for the quantifiers generalizing those suggested in [14, 37]. Covering an undecidable fragment of real arithmetic involving addition, subtraction, multiplication and transcendental functions, measuring solution sets exactly by an algorithm obviously is infeasible. The solving procedure therefore approximates the exact satisfaction probability of the formula under investigation and terminates with a conclusive result whenever the approximation gets tight enough to answer the question whether the satisfaction probability is above or below a target value specified by the user, e.g., a safety target.

We will subsequently introduce the logic manipulated by our solver, then explain the solving procedure, and finally demonstrate its use for analyzing stochastic hybrid automata.

The syntax of the input language of the solver, which is SSMT formulae over continuous quantifier domains, CSSMT for short, agrees with the discrete version from [15], except that continuous quantifier ranges are permitted.

Definition 1 An SSMT formula with continuous domain (CSSMT formula) is of the form $\Phi = \mathcal{Q} : \varphi$, where

- $\mathcal{Q} = Q_1 x_1 \in \mathrm{dom}(x_1) \ldots Q_n x_n \in \mathrm{dom}(x_n)$ is a quantifier prefix binding a sequence $x_1, \ldots, x_n$ of quantified variables. Here $\mathrm{dom}(x_i)$ denotes the domain of variable x_i, which may be an interval over the reals or integers. Each Q_i either is an existential quantifier $\exists$ or a randomized quantifier $\mathbb{R}_{p_i}$ assigning a computable probability density function p_i over $\mathrm{dom}(x_i)$ to x_i.[4]

[4]In practice, we offer a selection from a set of predefined density functions over the reals. For discrete carriers, we offer the ability to write arbitrary distributions by means of enumeration.

- φ is a quantifier-free formula over some arithmetic theory $\mathcal{T}$, in our case involving addition, subtraction, multiplication, exponentiation, and transcendental functions, as supported by the iSAT SAT-modulo-theory solver. As definitional translations [40] permit to rewrite arbitrary formulae to equisatisfiable conjunctive normal forms (CNF), we may w.l.o.g. assume that φ is in conjunctive normal form (CNF), i.e., that φ is a conjunction of clauses, and a clause is a disjunction of (atomic) arithmetic predicates of the forms $v \sim c$ or $v = e$, where v is a variable, $\sim \in \{<, \leq, =, \geq, >$ a relational symbol, c a rational constant, and e an expression over variables involving addition, subtraction, multiplication, and transcendental functions. φ is also called the *matrix*[5] of the formula.

The semantics of CSSMT formulae is defined by a $1\frac{1}{2}$-player game mediated by the alternation in the quantifier prefix: following the sequence in the quantifier prefix and respecting the possible moves permitted by the quantifier domains, an existential player tries to maximize the overall probability of satisfaction while her randomized opponent subjects the existential player's strategy to random disturbances. Formally, the semantics can be defined by a Bellman backward induction over the game graph [5], akin to the semantics for SSAT [33]:

Definition 2 The semantics of a CSSMT formula $\Phi = \mathcal{Q} : \varphi$ is defined by its probability of satisfaction $Pr(\Phi)$ as follows, where ε denotes the empty quantifier prefix:

$$
\begin{aligned}
Pr(\varepsilon : \varphi) &= 0, \text{ if } \varphi \text{ is unsatisfiable;} \\
Pr(\varepsilon : \varphi) &= 1, \text{ if } \varphi \text{ is satisfiable;} \\
Pr(\exists x_i \in \mathrm{dom}(x_i)\, \mathcal{Q} : \varphi) &= \sup_{v \in \mathrm{dom}(x_i)} Pr(\mathcal{Q} : \varphi[v/x_i]) \ ; \\
Pr(\mathfrak{R}_{p_i} x_i \in \mathrm{dom}(x_i)\, \mathcal{Q} : \varphi) &= \int_{\mathrm{dom}(x_i)} Pr(\mathcal{Q} : \varphi[v/x_i])\, p_i(v) \mathrm{d}v \ .
\end{aligned}
$$

Here, $\mathcal{Q}$ denotes an arbitrary (possibly empty) quantifier prefix and $\varphi[v/x_i]$ signifies the substitution of value v into φ.

According to Definition 2, the semantics yields the *supremal probability of satisfaction* $Pr(\Phi)$, which is computed by resolving the quantifiers from left to right, whereby existential quantifiers are resolved by an optimal strategy guaranteeing highest reward and randomized quantifiers yield the expectation of the remaining formula. For a quantifier-free formula, and thus also after all quantifiers have been resolved, the probability of satisfaction of the matrix ϕ is associated with its satisfiability.

Example 1 Figure 3 exemplifies the semantics by depicting a simplified image of the infinitely branching tree spanned by the domains of the individual quantifiers. Equivalent branches have been collapsed, which is signified by the intervals associated to branches in the graphics. The graphics shows the game tree constructed according to the semantics of CSSMT formula $\Phi = \exists x \in [-1, 1] \mathfrak{R}_{\mathcal{N}(\iota,\infty)} y \in (-\infty, +\infty) :$

[5] In SSAT parlance, this is the body of the formula after rewriting it to prenex form and stripping all the quantifiers.

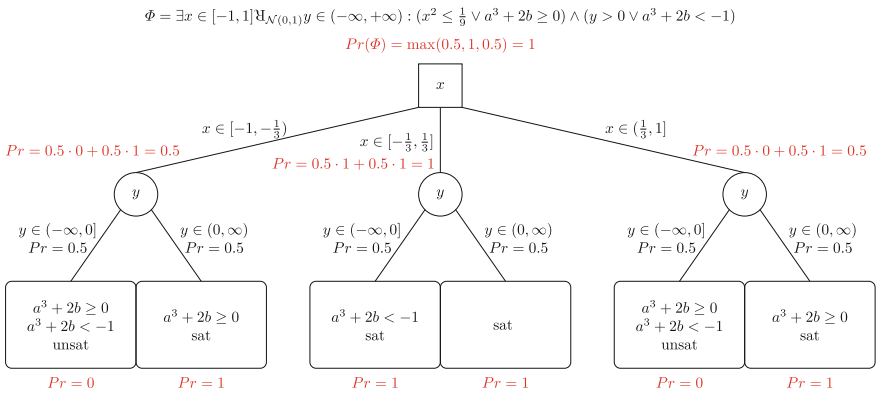


Fig. 3 Semantics of a CSSMT formula depicted as a quantifier tree with agglomerated branches

$(x^2 \leq \frac{1}{9} \vee a^3 + 2b \geq 0) \wedge (y > 0 \vee a^3 + 2b < -1)$, where $\mathcal{N}(0, 1)$ refers to the normal distribution with mean value 0 and variance 1. Semantically, Φ determines the maximum probability of the matrix across all values of x between $[-1, 1]$ when the values of y are distributed according to a standard normal distribution, i.e., determines an optimal choice of x as strategy for the existential player. We have grouped the uncountably infinitely many instances of the quantified variables into a finite set of branches reflecting cases not distinguished by the matrix. In the first branch, the domain of x is split into three parts, i.e., $x \in [-1, -\frac{1}{3})$, $x \in [-\frac{1}{3}, \frac{1}{3}]$ and $x \in (\frac{1}{3}, 1]$. For each part, we branch the domain of y into two parts. When all the quantified variables are resolved, we can check the satisfiability. For example, the leftmost leaf can be annotated with probability of satisfaction of 0, because $x \in [-1, -\frac{1}{3}]$ and $y \in (-\infty, 0]$ for this branch and the matrix consequently cannot be satisfied. When all the branches have been annotated, we can propagate the probability according to the corresponding quantifiers towards the root of the tree. For example, as y is distributed according to a standard normal distribution, the probability that $y \in (-\infty, 0]$ is 0.5. If we combine the probability from bottom to top and choose maximum value across the branches for x, as x is existentially quantified, then we obtain the probability of satisfaction of Φ, which in this simple case (but of course not generally) is 1. □

The Bellman backward induction inherent to Definition 2 seems to suggest building tool support based on branching the quantifier tree and calling appropriate SAT-modulo-theory (SMT) solvers on the (many) instances of the matrix thus evolving, as sketched in the example. This, however, is impractical for two reasons:

1. when continuous quantifier domains are involved, the number of branches to be spawned, and thus of SMT problems to be solved, would be uncountably infinite,[6]

[6]To this end please note that collapsing equivalent branches, as pursued in Fig. 3, can only be done *after* solving the instances of the matrix and thus only is an option in cases where continuity arguments (or similar) permit generalizations from samples to neighborhoods.

2. even in the case of merely discrete domains, the number of branches is strictly exponential in the quantifier depth and thus rapidly becomes prohibitive for the bounded model-checking (BMC) problems we want to solve, which feature a quantifier depth proportional to the depth of BMC.

We do consequently need more efficient means of solving CSSMT formulae, which are subject of the next section.

3.2 CSSMT Solving

We will now expose a practical algorithm for solving a CSSMT formula. As the exact probability $Pr(\mathcal{Q} : \varphi)$ of satisfaction is not computable in case the matrix φ stems from an undecidable fragment of arithmetic, as usual in the hybrid-system domain, we formulate the goal of solving as an approximate decision problem. The problem we want our solving engine to resolve therefore is formalized as follows:

Definition 3 Given a CSSMT formula $\Phi = \mathcal{Q} : \phi$, a reference probability $\varepsilon \in [0, 1]$, and a desired accuracy $\delta \geq 0$, a procedure which upon termination returns

- "GE", if $Pr(\Phi)$ is greater than or equal to $\varepsilon + \delta$;
- "LE", if $Pr(\Phi)$ is less than or equal to $\varepsilon - \delta$;
- "GE" or "Inconclusive", if $Pr(\Phi) \in [\varepsilon, \varepsilon + \delta]$;
- "LE" or "Inconclusive", if $Pr(\Phi) \in [\varepsilon - \delta, \varepsilon]$.

is called sound. It is called quasi-complete if it terminates whenever $\delta > 0$.

A sound and quasi-complete solver for CSSMT thus is required to yield a definite (and correct) answer whenever the actual probability of satisfaction is separated from the acceptance threshold ε by at least δ. If closer to the threshold than δ, it may provide inconclusive, yet never counter-factual answers.

Our method for solving CSSMT formulae is intuitively split into three distinct —yet overlapping in practice— phases:

1. *Quantifier branching*: In this phase, each quantified variable's domain is covered by a finite interval-partitioning, thereby branching a *collapsed* quantifier tree akin to that depicted in Fig. 3. In contrast to the case depicted in Fig. 3, we do, however, neither make sure that the individual multi-dimensional cells (i.e., product of intervals) thus obtained contain points indistinguishable by the matrix in the sense of all yielding the same truth value, nor use the same partitioning in all sub-trees. Due to the latter, we are not confined to regular gridding of the n-dimensional variable space, which helps in adaptive local refinement enhancing precision.
2. *Paving*: For each multi-dimensional cell C generated in the previous phase, we generate two sets of sub-boxes[7] of the cell: one set $\underline{C}$ under-approximating the

[7] As usual in interval constraint solving, we call any product of intervals with computer-representable bounds a *box*.

set of points $p \in C$ satisfying the matrix and another set $\overline{C}$ over-approximating the set of points $p \in C$ satisfying the matrix. Such covers can be obtained by established paving techniques from interval analysis, e.g., by using the RealPaver tool [20]. The sum of the —easily computable due to box shape— measures $\mu(B)$ of the boxes $B \in \underline{C}$ (or analogously $B \in \overline{C}$) provide a lower estimate l_C (upper estimate u_C, resp.) of the probability of satisfaction over C.
The measure $\mu(B)$ of a box $B = \prod_{i=1}^{n} [a_i, b_i]$, where n coincides with the number of quantified variables $x_1, \ldots, x_n$, thereby is defined as

$$\mu(B) = \prod_{i=1}^{n} \mu_i([a_i, b_i]), \text{ where}$$

$$\mu_i(X) = \begin{cases} \int_X \nu_n(x)\mathrm{d}x & \text{if } x_n \text{ is randomly quantified with density } \nu_n\,, \\ 1 & \text{if } x_n \text{ is existentially quantified.} \end{cases}$$

Note that this measure does not expose an effect of existential quantification, as B is an axis-parallel box such that the choices of existential variables impose no constraints on the random variables.

3. *Projection and lifting*: Given the bounds l_C and u_C for the satisfaction probability over each cell C, the final phase recursively synthesizes the overall satisfaction probability over the full domain by combining cells according to the quantifier prefix. If C and D are (necessarily neighboring) cells together forming a larger box-shaped cell (i.e., a product of intervals) E, then their probability masses can be combined as follows: if l_C, l_D are the respective lower and u_C, u_D the respective upper estimates of the satisfaction probabilities, and if C and D are adjacent in direction of variable x, then

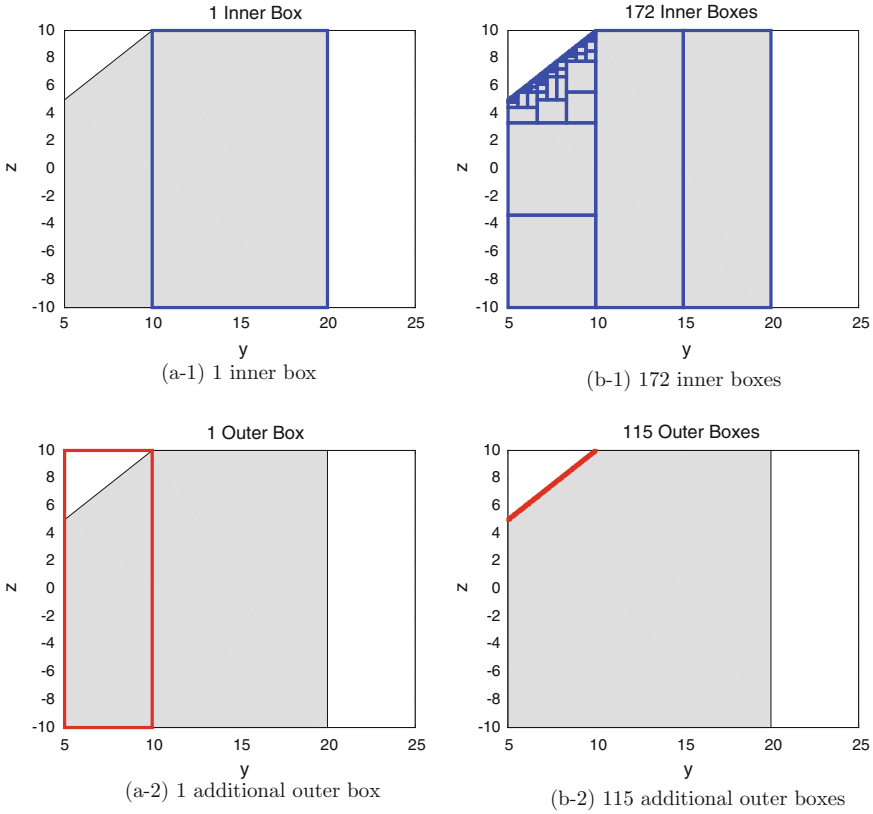
$$l_E = \begin{cases} \max\{l_C, l_D\} & \text{if } x \text{ is existentially quantified,} \\ l_C + l_D & \text{if } x \text{ is randomly quantified,} \end{cases}$$

$$u_E = \begin{cases} \max\{u_C, u_D\} & \text{if } x \text{ is existentially quantified,} \\ u_C + u_D & \text{if } x \text{ is randomly quantified.} \end{cases}$$

In practice, these three phases should, however, be interleaved in order to provide adaptive refinement of covers in quantifier branching. That is, new cells are generated —and thus, the quantifier tree from phase 1 expanded— if and only if the computation of basic set measures in phase 2 or the computation of combined measures in phase 3 yield overly large differences between lower (l_C) and upper (u_C) probability estimates for some cell. In that case, the cell will be split into two in order to facilitate a sharper approximation of the actual satisfaction probability within the cell. Vice versa, local estimates generated in phase 3 can be used for pruning expansion of the quantifier tree from phase 1, generalizing Teige's effective search-space reduction [37] to the CSSMT case. To this end, it should be noted that residual cells need not be investigated if their total probability mass does not suffice to lift a partially

computed (due to phase 2) mass above the acceptance threshold ε, or if the threshold already is exceed even without their contribution, or if alternative branches can be decided to yield better reward, to name just a few of numerous pruning rules rendering the procedure computationally tractable. The interested reader may refer to [37] for details of such pruning rules.

After generating an upper estimate u and a lower estimate l for the whole domain, all that remains is to check for their relation to the threshold ε: as u is a safe upper approximation of $P(\phi)$, we can report "LE", i.e., $P(\phi) \leq \varepsilon$ whenever $u \leq \varepsilon$. Similarly, as l is a safe lower approximation of $P(\phi)$, we can report "GE", i.e., $P(\phi) \geq \varepsilon$ whenever $l \geq \varepsilon$. If, however, $l < \varepsilon < u$ then the test is inconclusive, which we are



(a-1) 1 inner box

(b-1) 172 inner boxes

(a-2) 1 additional outer box

(b-2) 115 additional outer boxes

Fig. 4 Two-dimensional projections of the inner and outer approximations for the solution set of matrix $x > 3 \wedge y \leq 20 \wedge x^2 > 49 \wedge y \geq z$ over computation cell $x \in [7, 10]$, $y \in [5, 25]$ and $z \in [-10, 10]$ (corresponding to outermost frame). The *gray area* marks the exact set of models of the matrix within the cell. The *left side* gives a very rough approximation of the solution set by one inner box (a-1) and two outer boxes (a-1 plus a-2) for the over- and under-approximation, respectively. The *right side* provides a much tighter inner (b-1) and outer (b-1 plus b-2) approximation by 172 and 287 boxes, respectively

allowed to report iff $\varepsilon - l < \delta$ and $u - \varepsilon < \delta$. If the latter is not the case then we have to refine the cover of the quantifier domains by computation cells, which we obviously do by splitting those cells having the highest difference between their upper and lower probability estimates, i.e., the ones featuring the lowest accuracy.

Example 2 We consider the paving phase for the CSSMT formula

$$\begin{aligned}\Phi = \exists\, x \in [-10, 10]\, \mathfrak{X} y \in \mathcal{U}[5, 25]\, \mathfrak{X} z \in \mathcal{U}[-10, 10] : \\ (x > 3 \vee y < 1) \wedge (z > x^2 + 2 \vee y \leq 20) \wedge \\ (x^2 > 49 \vee y > 7x) \wedge (x < 6 \vee y \geq z)\ ,\end{aligned}$$

where $\mathcal{U}[a, b]$ refers to uniform distribution with carrier $[a, b]$. If the paving procedure generates just one box for the under- and two for the over-approximation, as shown in Fig. 4(a-1 and a-2), the measures returned are $l = 0.5$ and $u = 0.75$. A more precise estimation will be obtained by generating more boxes in the paving: the 172 inner boxes and 287 outer boxes of Fig. 4(b-1 and b-2) yield $l = 0.7181$ and $u = 0.7191$. □

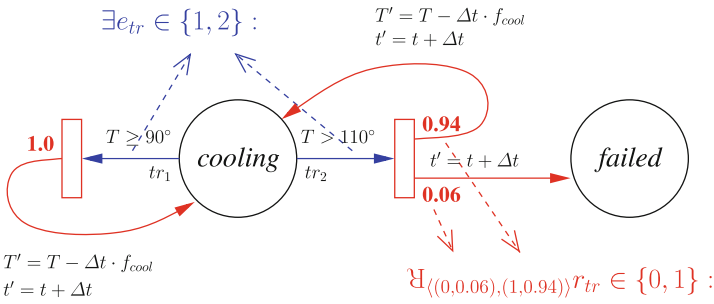
3.2.1 Encoding Bounded Reachability for SHA

CSSMT solving permits bounded model checking (BMC) of stochastic hybrid automata just the same way SMT facilitates it for hybrid automata (HA) devoid of stochasticity. The encodings of the transition relation are virtually identical to those established for HA [3, 12, 23], yet the alternation between non-deterministic and stochastic branching additionally has to be encoded by a corresponding alternation of $\exists$ and $\mathfrak{X}$ quantifiers. The basic idea is illustrated in Fig. 5.

For computing the worst-case (in the sense that non-deterministic choices are resolved by a malicious adversary) probability of reaching a bad state in k steps, where bad states are encoded by a predicate Bad, this encoding is unwound to a formula Φ of the shape

$$\underbrace{\exists t_1\, \mathfrak{X}_d\, p_1 \exists t_2\, \mathfrak{X}_d\, p_2 \ldots \exists t_k\, \mathfrak{X}_d\, p_k}_{\text{alternating non-determinist. and probabil. choices}} : \underbrace{\underbrace{\begin{pmatrix} Init(\vec{x}_0) \\ \wedge\, Trans(\vec{x}_0, \vec{x}_1) \\ \wedge\, Trans(\vec{x}_1, \vec{x}_2) \\ \wedge \ldots \\ \wedge\, Trans(\vec{x}_{k-1}, \vec{x}_k) \end{pmatrix}}_{k\text{-bounded reach set}} \wedge \underbrace{\begin{pmatrix} Bad(\vec{x}_0) \\ \vee\, Bad(\vec{x}_1) \\ \vee\, Bad(\vec{x}_2) \\ \vee \ldots \\ \vee\, Bad(\vec{x}_k) \end{pmatrix}}_{\text{hits bad state}}}_{\text{BMC}(k)},$$

where the k-fold alternating quantifier prefix reflects the alternation between non-deterministic and randomized choices inherent to the semantics of SHA and where the matrix is a conventional symbolic encoding of the k-step BMC problem. The



source $\wedge$ guard	$\wedge$ transition selection	$\wedge$ random distrib.	$\wedge$ action	$\wedge$ target
$\Big(cooling \wedge (T \geq 90°)$	$\wedge\ (e_{tr} = 1)$	$\wedge$ true	$\wedge \begin{pmatrix}(T' = T - \Delta t \cdot f_{cool}) \\ \wedge\ (t' = t + \Delta t)\end{pmatrix}$	$\wedge\, cooling'\Big) \vee$
$\Big(cooling \wedge (T > 110°)$	$\wedge\ (e_{tr} = 2)$	$\wedge\ (r_{tr} = 0)$	$\wedge\ (t' = t + \Delta t)$	$\wedge\, failed'\Big) \vee$
$\Big(cooling \wedge (T > 110°)$	$\wedge\ (e_{tr} = 2)$	$\wedge\ (r_{tr} = 1)$	$\wedge \begin{pmatrix}(T' = T - \Delta t \cdot f_{cool}) \\ \wedge\ (t' = t + \Delta t)\end{pmatrix}$	$\wedge\, cooling'\Big)$

Fig. 5 Principle of encoding stochastic hybrid automata: non-deterministic choice maps to existential quantification (*top left part* of the graphics), probabilistic choice to randomized quantification (*bottom right part* of the graphics), and the transition relation is encoded symbolically as in SMT-based BMC (table), yet adding the dependencies on choices (columns 3 and 4 in the formula)

quantifiers and the symbolic transition relation $Trans$ correspond to the respective objects in Fig. 5. Details of this encoding can be found in [15, 16, 37]. Its central property is that the satisfaction probability $Pr(\Phi)$ of the resulting formula is exactly identical to the worst-case probability of the encoded SHA reaching a bad state within k steps, such that CSSMT solving can be used for discharging the proof obligations arising from bounded probabilistic reachability problems. Verification problems of the form "can the the (worst-case under all adversary strategies resolving non-determinism) probability of reaching a bad state over a horizon of k steps be guaranteed to stay below a given safety target ε" are thus amenable to automatic analysis.

It should be noted that the above encoding yields extremely deep quantifier prefixes, as the alternation depth grows linearly in the number of steps of the probabilistic BMC problem. While this might seem to render (C)SSMT solving infeasible, adequate pruning rules in SSMT proof search permit to solve surprisingly large instances [37], speeding up solving by up to significantly more than ten orders of magnitude compared to naïve quantifier traversal.

4 Parameter Synthesis for Parametric Stochastic Hybrid Automata

Within the model of stochastic hybrid automata (SHA), we allowed for stochastic updates on the continuous as well as discrete variables using fixed probability distributions. This enables modeling of, among many other random phenomena, random component failures or data packet losses. Additionally, SHA contained non-determinism in terms of non-deterministic values and transitions. Extending SHA, within this section, we allow for an additional non-determinism in terms of a parametric dependence of the discrete probabilistic branching by replacing transition probabilities with parametric terms (t_i in Fig. 6). However, apart from this parametric non-determinism within the discrete probabilistic transitions, we require the system to be either deterministic or probabilistic, hence reducing the expressive power of SHA. Therefore, the resulting *parametric stochastic hybrid automata* (PSHA) feature a finite set of discrete locations (or modes), each of which comes decorated with a differential equation governing the dynamics of a vector of continuous variables while residing in that mode.

Modes change through instantaneous transitions guarded by conditions on the current values of the continuous variables, and may yield discontinuous (potentially stochastic) updates of the continuous variables. Aiming at simulation-based

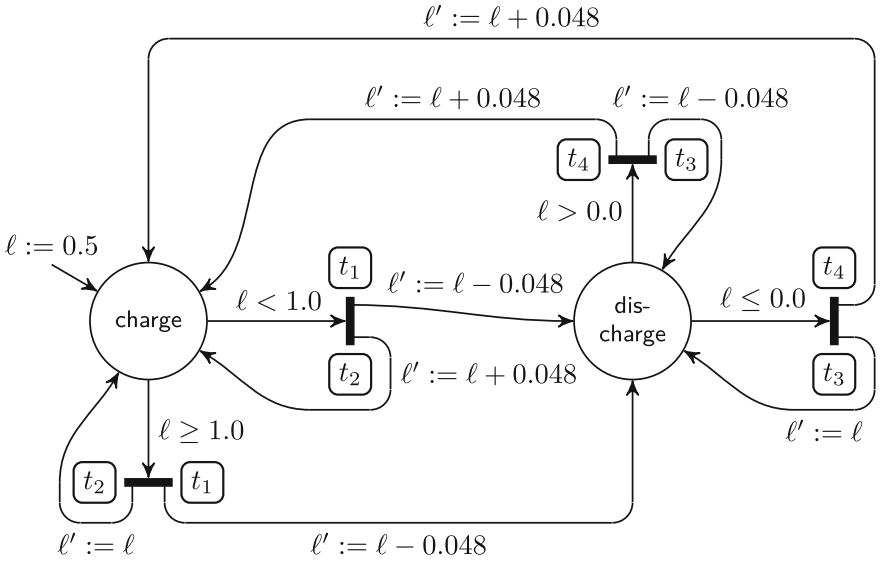


Fig. 6 PSHA model of a charging station. Modes are labeled with labels `charge` and `discharge` abbreviating ODE (not shown explicitly) representing corresponding dynamics over a continuous capacity ℓ. Modes can switch according to guarded transitions leading to a probabilistic branch. Probabilities are summarized as terms $t_1, \ldots, t_4$ indicating their parameter dependencies

evaluation methods as in Statistical Model Checking (SMC) [42], transition selection in this section is assumed to be deterministic, i.e., guard conditions at each mode are mutually exclusive or overlaps are resolved deterministically as, e.g., by the priority mechanisms in Simulink-Stateflow. To prevent non-determinism between possible time flows and transitions, we also assume that transitions are urgent, i.e., they are taken as soon as they are enabled (which furthermore renders mode invariants redundant). In addition to these mechanisms from deterministic hybrid automata, PSHA allow for the probabilistic selection of a transition variant based on a discrete random experiment. The probability distribution governing the random experiment is allowed to have a parametric dependence. Following the idea of Sproston [35, 36], the selected transition entails a randomized choice between transition variants according to a discrete probability distribution. The different transition variants can lead to different follow-up locations and different continuous successors, as depicted in Fig. 6, where the guard condition determining transition selection is depicted along the straight arrows leading to a potential branching annotated with probability terms denoting the random experiment.

To model the parametric dependence of PSHA, we allow the branching mode-transition probabilities to be terms over a set Θ of parameter names (t_i in Fig. 6). The viable parameter instances $\theta : \Theta \to \mathbb{R}$ are constrained by an arithmetic first-order predicate ϕ over Θ, defining their mutual relation. Let $\Phi = \{\theta : \Theta \to \mathbb{R} \mid \theta \models \phi\}$ denote the set of all viable parameterizations. Arithmetic terms over Θ are subject to the constraint that for all viable parameter valuations $\theta \models \phi$, the sum of outgoing probabilities assigned to each transition is 1, i.e., $\phi \implies \sum_{i=1}^{n} t_i(\theta) = 1$ holds for the probability terms t_i associated to each transition t. Note that the probability terms need not contain free variables from θ; non-parametric distributions are special cases of parametric distributions and do not require special treatment.

For the sake of formal analysis, we formalize the semantics of PSHA through a reduction to a parametric infinite-state Markov chain. For a PSHA with location set Λ and continuous variables $x_1, \ldots, x_D$, the states of the Markov chain are given by $\Sigma = \Lambda \times \mathbb{R}^D$ and the initial state distribution is inherited from the PSHA. Each state $\sigma = (l, \vec{x}) \in \Sigma$ gives rise to a parameter-dependent probability distribution of successor states σ':

$$p_\sigma(\sigma', \theta) = \begin{cases} t(\theta) & \text{if a transition } (\sigma, \sigma') \text{ labeled with probability term } t \text{ is enabled in } \sigma, \\ 1 & \text{if } \sigma' = (l, g(t)), \text{ where } g \text{ is a solution to the ODE associated to } l \in \Lambda \text{ with } g(0) = \vec{x}, \text{ and no transition is enabled in } (l, g(t')) \text{ for any } t' \in [0, t[, \text{ and a transition is enabled in } \sigma' = (l, g(t)), \\ 0 & \text{otherwise.} \end{cases}$$

Given a parametric infinite-state Markov chain M with its initial state distribution given by a density $\iota : \Sigma \to \mathbb{R}_{\geq 0}$ and a parametric next-state probability mass function $p_\sigma : \Sigma \times \Phi \to [0, 1]$, the distribution associated to finite runs $\langle \sigma_0, \sigma_1, \ldots, \sigma_k \rangle \in \Sigma^*$ given a parameter instance $\theta \in \Phi$ is given by the following probability mass function:

$$p_M(\langle \sigma_0, \sigma_1, \ldots, \sigma_k \rangle; \theta) = \iota(\sigma_0) \cdot \prod_{i=0}^{k-1} p_{\sigma_i}(\sigma_{i+1}, \theta).$$

Note that θ can be vector valued, comprising multiple individual parameters.

Let $f : \Sigma \to \mathbb{R}$ be a scalar function on states, to be evaluated on the last state of a run and called the *reward* f of the run,[8] and let $k \in \mathbb{N}$. The *k-bounded expected reward for f in a parameter instance $\theta \in \Theta$* is

$$\mathcal{E}_{M,k}[f; \theta] = \int_{\Sigma^k} f(\sigma_{k-1}) p_M(\langle \sigma_0, \sigma_1, \ldots, \sigma_{k-1} \rangle; \theta) \, \mathrm{d}\langle \sigma_0, \sigma_1, \ldots, \sigma_{k-1} \rangle, \quad (1)$$

where Σ^k denotes the sequences over Σ of length k. We will subsequently drop the index M in $\mathcal{E}_{M,k}$ and p_M whenever it is clear from the context. Although the finite nature of the parametric-dependent probabilistic branching would allow us to write the expectation in terms of a sum, we represent the expected value in form of an integral in Eq. (1) to illustrate the similarities to SHA and the general applicability of importance sampling (see below).

Rewards represent quantitative measures of the system's performance, and therefore mutual constraints on their values can be used for capturing design goals. The design problem we are thus facing is, given a vector $f_1, \ldots, f_n : \Sigma \to \mathbb{R}$ of rewards in a parametric infinite-state Markov chain M, to ensure via adequate instantiation of the parameter that the expected rewards meet the design goal:

Definition 4 (*Parameter synthesis problem*) Let $f_1, \ldots, f_n : \Sigma^k \to \mathbb{R}$ be a vector of rewards in a parametric Markov chain M and let C be a *design goal* in the form of a constraint on the expected rewards, i.e., an arithmetic predicate containing $f_1, \ldots, f_n$ as free variables. A parameter instance $\theta : \Theta \to \mathbb{R}$ is *feasible (w.r.t. M and C)* iff

$$\theta \models \phi \quad \text{and} \quad [f_1 \mapsto \mathcal{E}(f_1; \theta), \ldots, f_n \mapsto \mathcal{E}(f_n; \theta)] \models C.$$

The *multi-objective parameter synthesis problem* is to find a feasible parameter instance θ, if it exists, or to prove its absence otherwise.

Stated in words, a parameter instance θ is feasible w.r.t. ϕ and C iff the parameters are in the range defined by ϕ and the expected rewards resulting from the instantiation of M with θ meet the multi-objective C. Note that the aim is to find some parameter

[8] Due to the generality of the PSHA model, defining rewards exclusively on the final state is as expressive as defining them via functions on the whole run.

instance meeting our design goal; we are not considering determining the set of all suitable instantiations. That is, in contrast to the setting in the previous Sect. 3, we are here interested in a parameter value satisfying the constraints on multiple rewards rather than determining the probability of satisfaction for a given parameter instance. In fact, the constraint system of Definition 4 is indeed more general, as one can specify as objective a constraint system relating different expected values rather than just a required threshold on a single probability. For example, with the notation in Definition 4, it is possible to ask for a parameter instance, such that one expected value is larger than another expected value, both of which are allowed to depend on the parameter value under consideration. Note that such a problem cannot be stated using the CSSMT formalism of Sect. 3. On the other hand, the technology from Sect. 3 can deal with $1\frac{1}{2}$-player games, i.e., SHA involving both stochasticity and non-determinism, whereas we here deal with probabilistic systems only — albeit parametric ones.

4.1 Parameter Synthesis Using Symbolic Importance Sampling

Expected values aggregate contributions from many different states or trajectories as in Eq. (1). In particular, when each of the states contributes a different non-zero value to the overall expected values, an exact calculation of the expected value requires to evaluate all states or trajectories. As a result, the computation of the expected values can be the main bottleneck in finding a suitable parameter instance. We aim in the following at a statistical evaluation of the expected value, which scales with the number of samples used for such a statistical evaluation and therefore has the potential of producing results faster. As these statistical estimates of the expected values are merely arithmetic expressions, we then use these expressions to construct a constraint system which represents the parameter-dependencies of the expectations symbolically and can be solved using available constraint solvers, such as the iSAT solver [14]. However, due to the statistical sampling underlying the generation of the constraint system, these results are also only of statistical nature. That is, instead of finding a parameter instance for which we can rigorously guarantee that the constraint system is satisfied, we can only guarantee with a well-defined statistical confidence that the constraint system is likely to be satisfied. Similarly, we can only bound the probability that no parameter instance exists satisfying the constraint system in case of a negative satisfiability result.

In [18], we developed a scheme which uses a symbolic version of importance sampling in order to use a sampling-based strategy for estimating expected values while keeping track of the parametric dependence of these expectations, which we will review in the following. This technique combines statistical model-checking of a parameter-free instantiation of the parametric hybrid system with a symbolic

variant of importance sampling in order to learn a symbolic model of the parameter dependency.

In order to introduce the general concept of importance sampling [39], we mostly abstract from our PSHA setting in this section, but highlight specifics to the PSHA setting when necessary. We instead assume that the parametric probability distribution of the random variable $x \in X$ is given in terms of a density function $p(\cdot; \theta)$ which depends on a vector θ of bounded real-valued parameters. Permissible values of θ are defined by a first-order constraint ϕ.

Given an arbitrary (bounded) function $f : X \to \mathbb{R}$, we are interested in estimating expected values of f under all parameter values $\theta \models \phi$. The expectation $\mathcal{E}[f; \theta]$ for reward f given parameter vector θ is

$$\mathcal{E}[f; \theta] = \int_X f(x)\, p(x; \theta)\, \mathrm{d}x \ . \tag{2}$$

Given a specific parameter instance θ^* and a process sampling x_i according to the distribution $p(\cdot; \theta^*)$, the expectation $\mathcal{E}[f; \theta^*]$ can be estimated by

$$\tilde{\mathcal{E}}[f; \theta^*] = \frac{1}{N} \sum_{i=1}^{N} f(x_i) \ , \tag{3}$$

which is the empirical mean of the sampled values of reward f. In our PSHA setting, a reasonable process for generating such samples x_i according to the distribution $p(\cdot; \theta^*)$ would be a simulator for non-parametric SHA, applied to the instance of the PSHA under investigation obtained by substituting concrete parameter values θ^* for the free parameters.

For sufficiently large N, we expect $\mathcal{E}[f; \theta^*] \approx \tilde{\mathcal{E}}[f; \theta^*]$ due to the law of large numbers. We can quantify the quality of the approximation in (3) using Hoeffding's inequality [24], provided that f has a bounded support $[a_f, b_f]$:

$$\begin{aligned} P\left(\mathcal{E}[f; \theta^*] - \tilde{\mathcal{E}}[f; \theta^*] \geq \varepsilon\right) &\leq \exp\left(-2\frac{\varepsilon^2 N}{(b_f - a_f)^2}\right) \ , \\ P\left(\tilde{\mathcal{E}}[f; \theta^*] - \mathcal{E}[f; \theta^*] \geq \varepsilon\right) &\leq \exp\left(-2\frac{\varepsilon^2 N}{(b_f - a_f)^2}\right) \ . \end{aligned} \tag{4}$$

Therefore, the empirical mean (3) yields a very reliable estimate of the actual expectation when the number of samples is large, with the accuracy given by (4).

As can be seen from Eq. (3), the estimate of the expected value depends only implicitly on the parameter of interest as the samples x_i are drawn from a fixed probability distribution using the concrete parameters θ^*. Consequently, one has to fix some parameter instance to generate the samples x_i, thereby losing the parametric

dependence. To alleviate this problem, we one can use importance sampling using an arbitrary proposal distribution q, which does not depend on any parameter. Specifically, one can use the same sampling approach as (3) for (5), however, modifying the reward function:

$$\mathcal{E}[f;\theta] = \int_X f(x)\frac{p(x;\theta)}{q(x)}q(x)\,\mathrm{d}x \tag{5}$$

Using the same naïve Monte Carlo estimate yields the following empirical approximation to the expectation, including the parameter dependence on θ using N samples drawn from the substitute distribution q:

$$\tilde{\mathcal{E}}_q[f;\theta] = \frac{1}{N}\sum_{i=1}^{N} f(x_i)\frac{p(x_i;\theta)}{q(x_i)} \ . \tag{6}$$

Note that such a procedure can be used to obtain unbiased estimates of the expectation for both continuous probability distributions (densities) as well as discrete probability distributions (probability mass functions).

Doing so, however, requires being able to actually compute the quotient $\frac{p(x;\theta)}{q(x)} =: gain(x_i, \Theta)$ for each sample x_i. Whenever x_i is a trace in a PSHA, this can easily be achieved by taking

$$gain(x_i, \Theta) = \prod_{i=1}^{k}\left(\frac{t_i(\Theta)}{t_i(\Theta^*)}\right)^{\#t_i(x_i)} \ , \tag{7}$$

where $t_1, \ldots, t_k$ are the different parametric terms occurring in the automaton and $\#t_i(x_i)$ is the number of times the transition marked with t_i was taken in trace x_i. Note that $gain(x_i, \Theta)$ contains only the parameter vector Θ as free variables (all other entities are constants), such that (6) provides a *symbolic* expression for the parameter-dependency of $\mathcal{E}[f;\theta]$. For details concerning this automatically generated symbolic encoding, the interested reader may confer [18].

To synthesize a feasible parameter instance, we can generate an arithmetic constraint characterizing feasibility by plugging the symbolic parametric estimate (6) with the concrete gain term (7) into the condition for parameter feasibility from Definition 4, thereby adding an additional constraint for error control in the second line of the following equation. Here, $\epsilon(q_i, \delta, N)$ is an uncertainty term which captures the variability of the estimates as a function of the proposal distribution, the number of samples, and the confidence.

$$\begin{aligned} \theta \models \phi \quad &\text{and} \quad [f_1 \mapsto \mathcal{E}(f_1;\theta), \ldots, f_n \mapsto \mathcal{E}(f_n;\theta)] \models C \\ &\text{and} \quad \mathcal{E}(f_i;\theta)] \in [\tilde{\mathcal{E}}_{q_i}[f_i;\theta] - \epsilon(q_i,\delta,N), \tilde{\mathcal{E}}_{q_i}[f_i;\theta] + \epsilon(q_i,\delta,N)] \end{aligned} \tag{8}$$

Note that here, each of the expected values is replaced by the empirical estimate using samples drawn from a proposal distribution stemming from a parameter substitution Θ^*. As the resulting constraint system contains only constraints over arithmetic expressions, this system can directly be fed into a constraint solving engine such as iSAT [14], which is able to efficiently handle the polynomials of high degree stemming from the gain term (7). Due to the randomness involved in sampling, the estimates are themselves random variables. The result (a parameter instance or an infeasibility result) thus itself is subject to random fluctuations and we can guarantee the correctness of this result only with probability $\geq 1 - \delta$.

In order for this result to be valid, we have to determine $\epsilon(q_i, \delta, N)$ such that we can guarantee (with high probability)

$$\mathcal{E}(f_i;\theta) \in [\tilde{\mathcal{E}}_{q_i}[f_i;\theta] - \epsilon(q_i,\delta,N), \tilde{\mathcal{E}}_{q_i}[f_i;\theta] - \epsilon(q_i,\delta,N)] \ .$$

Importantly, due to the parameter dependence of the empirical estimates, one cannot use Hoeffding's inequality (4), but this statement has to hold uniformly across all θ satisfying ϕ.

Example 3 To illustrate this problem, consider the following example. Let $f(x) = \text{sign}(x - \pi)$ be a reward function on a continuous variable $x \in [0, 2\pi] \subset \mathbb{R}$. Further, let the parametric probability density $p(x;\theta) \propto (\sin(x\theta) + 1)$. When sampling x_i and calculating the empirical average reward, one can tune the corresponding Eq. (6) arbitrarily, such that the density $p(x;\theta)$ close to zero for all x_i with $f(x_i) = 1$ and close to one for all x_i with $f(x_i) = -1$ by appropriately choosing θ. However, the true expected reward is almost independent of θ. Importantly, for this choice of density, setting sampling points to zero or one is possible for arbitrarily many sampling points. Therefore, the empirical parametric expression does not necessarily converge to the true expectation with an increasing number of samples (as would be suggested by Hoeffding's inequality). To adjust for this effect, one has to account for the complexity of the parametric function. □

In fact, describing the effect of tuning parameters within such empirical expressions is one of the major research questions within the field of statistical learning theory (see [8, 41]), resulting in different complexity measures. For example, the sinusoidal function above has Vapnik-Chervonenkis dimension of infinity, indicating a very high complexity. By tuning the parameters after the data has been observed, a common phenomenon is called over-fitting, i.e., overly adapting the parameter to the data thereby introducing a larger error in the statistical estimate of the true value of the expected reward.

When using Eq. (8) together with a constraint solver as iSAT, we have to show the validity of the results, i.e., we have to show that the probability of the obtained result

being wrong is bounded by a pre-specified value δ. To this end, we first consider the case that the constraint solver produces the result UNSAT. That is, the constraint solver cannot find a candidate solution θ^* satisfying the constraint system (8). The reason for such unsatisfiability can either rightfully lie within infeasibility of the synthesis problem itself, or can be erratic due to the statistical uncertainty within the estimation of the expected values.

UNSAT Case

In case the solver cannot find a candidate value for the parameter such that the constraints are satisfied, we would like to bound the probability for this statement being wrong, i.e., an artifact of the randomness in sampling. To bound this probability, we can examine the following events for arbitrary probability thresholds c:

$$E_1 : \min_\theta \mathcal{E}[f_i;\theta] < c \quad \text{and} \quad \min_\theta \tilde{\mathcal{E}}[f_i;\theta] \geq c + \epsilon \tag{9}$$

$$E_2 : \max_\theta \mathcal{E}[f_i;\theta] > c \quad \text{and} \quad \max_\theta \tilde{\mathcal{E}}[f_i;\theta] \leq c - \epsilon \tag{10}$$

Intuitively, we would like to bound the probability that we were not able to solve a slightly easier task $\tilde{\mathcal{E}}[f_i;\theta] \geq c + \epsilon$ while the original task is possible $\mathcal{E}[f_i;\theta] < c$.

Theorem 1 (Confidence for UNSAT)
Let $\epsilon = 2B_i\sqrt{-\frac{\log(\delta)}{N}}$ *and* $B_i = \max_{x,\theta} \frac{f_i(x)p(x;\theta)}{q_i}$ *be given. Then* $P(E_1) \leq \delta$ *and* $P(E_2) \leq \delta$.

Proof For E_1 the following holds:

$$\begin{aligned}
&P\left(\min_\theta \tilde{\mathcal{E}}[f_i;\theta] \geq \epsilon + c \ \wedge \ \min_\theta \mathcal{E}[f_i;\theta] < c\right) \\
&\leq P\left(\min_\theta \tilde{\mathcal{E}}[f_i;\theta] \geq \min_\theta \mathcal{E}[f_i;\theta] + \epsilon\right) \\
&\overset{\text{Jensen ineq.}}{\leq} P\left(\underbrace{\min_\theta \tilde{\mathcal{E}}[f_i;\theta]}_{=:g(x_1,\ldots,x_N)} - \mathcal{E}\left[\min_\theta \tilde{\mathcal{E}}[f_i;\theta]\right] \geq \epsilon\right) \\
&= P\left(g(x_1,\ldots,x_N) - \mathcal{E}\left[g(x_1,\ldots,x_N)\right] \geq \epsilon\right) \\
&\overset{\text{McDiarmid ineq.}}{\leq} \exp\left(-\frac{\epsilon^2 N}{4B_i^2}\right) = \delta;\ B_i = \max_{x,\theta} \frac{f_i(x)p(x;\theta)}{q_i(x)}
\end{aligned} \tag{11}$$

To apply McDiarmid's inequality [31] in Eq. (11), we used the following bounds:

$$
\begin{aligned}
-\frac{2B}{N} &\leq \min_{\theta}\left\{\frac{1}{N}f(x_N;\theta)-\frac{1}{N}f(x'_N;\theta)\right\}\\
&\leq \min_{\theta}\left\{\frac{1}{N}\sum_{i}^{N-1}f\,(x_i;\theta)+\frac{1}{N}f(x_N;\theta)\right\}\\
&\quad -\min_{\theta}\left\{\frac{1}{N}\sum_{i}^{N-1}f\,(x_i;\theta)+\frac{1}{N}f(x'_N;\theta)\right\} \qquad (12)\\
&= \big(g(x_1,\ldots,x_N)-g(x_1,\ldots,x'_N)\big)\\
&\leq -\min_{\theta}\left\{\frac{1}{N}f(x'_N;\theta)-\frac{1}{N}f(x_N;\theta)\right\}\leq 2\frac{B}{N}\\
\Rightarrow &\left|\big(g(x_1,\ldots,x_N)-g(x_1,\ldots,x'_N)\big)\right|\leq 2\frac{B}{N}
\end{aligned}
$$

In general, due to the concavity of the minimum, we have

$$\min(f-g)\leq\min(f)-\min(g)\leq-\min(-(f-g))=\max(f-g) \qquad (13)$$

For E_2 the proof is analogous replacing min with max. □

SAT Case

Unfortunately, we are not able to construct a similar argument for the SAT case, as this would require calculating a complexity measure such as a Vapnik-Chernovenkis or Rademacher complexity for the function $\frac{f_i(x)}{q_i(x)}p(x;\cdot)$, for which the dependency on the parameter has to be analyzed to much more detail. However, if we construct the proposal distribution q_i by choosing a particular value of θ, e.g., $q_i(x)=p(x;\theta^*)$, we can use the standard Hoeffding inequality (4) to check whether this particular instance of parameter value θ^* satisfies the constraint system. That is, if we found that the constraint system (8) with the particular setting:

$$\tilde{\mathcal{E}}_{q_i}[f_i;\theta]=\frac{1}{N}\sum_{l}f_i(x_l)\ \text{ with }\ x_l\sim q_i=p(\cdot;\theta^*)\ \text{ and }\ \epsilon(q_i,\delta,N)=2\sqrt{-\frac{\log(\delta)}{N}}$$

is satisfied for θ^*, then we know that the original constraint system is also satisfied with a probability $\geq 1-\delta$, due to Hoeffding's inequality.

Taken the results obtained so far, we have the following algorithm for checking a constraint system involving expected values of parametric probability distributions:

1. Select a particular parameter instance $\theta^*\models\phi$.
2. Draw N samples $\{x_i\}$, $i=1,\ldots,N$ from the proposal distribution $q=p(\cdot;\theta^*)$.
3. Check the particular parameter instance for satisfaction of C using the empirical estimates $\tilde{\mathcal{E}}[f_i;\theta^*]=\frac{1}{N}\sum_k f_i(x_k)$ (see also (3)). To do so, we check if $\tilde{\mathcal{E}}[f_i;\theta^*]\pm 2\sqrt{-\frac{\log(\delta)}{N}}$ satisfies C. This step is referred to in Algorithm 1 as CHECKSAMPLES.

4. If the system is satisfied, we have found a feasible parameter instance θ^* with confidence $1-\delta$, i.e., the probability of violating the constraints on the expected rewards is smaller than δ. This is due to Hoeffding's inequality (4), which guarantees the high probability (see also [44] where the same method is applied).
5. If the system is unsatisfiable, we construct the empirical constraint system using (8).
6. Check the corresponding constraint system using iSAT once again.
7. If the system is unsatisfiable, we know with confidence $1-\delta$ that the original constraint system is unsatisfiable.

It could, however, happen that the first check yields unsatisfiable, while the second yields satisfiable, i.e., iSAT could neither verify the particular parameter instance θ^*, nor could refute existence of feasible parameter instances. In this case, the second check, however, generates another candidate parameter vector θ^{**} using the empirical parametric constraint system (8). With the newly obtained parameter instance, we can now re-iterate the algorithm until we either find a statistically valid solution or refute existence of feasible instances. Taken together, we obtain the iterative Algorithm 1.

Algorithm 1 Parameter Fitting by Symbolic Importance Sampling

function SYM- IMP(ϕ, C, confidence δ, number of samples N, max. iterations I)
  $\delta^c \leftarrow \frac{\delta}{I}$; $\theta_0 \leftarrow$ SOLVECONSTRAINTSYSTEM(ϕ); $\varepsilon \leftarrow \sqrt{\frac{\log(\frac{1}{\delta^c})}{N}}2B$; $m \leftarrow 0$; $\hat{\phi}_0 \leftarrow \phi$
  while $m \leq I$ **do**
    $q \leftarrow p(\cdot;\theta_m)$
    $S = (x_1, \ldots, x_N) \leftarrow$ DRAWSAMPLES(q, N) ▷ Simulate N times w. ▷ parameterization θ_m.
    if CHECKSAMPLES(S, δ, ϕ, C) **then**
      return θ_m ▷ Found parameterization satisfying C with prob. $\geq 1-\delta$
    else
      $\hat{\phi}_{m+1} \leftarrow \hat{\phi}_m \wedge \bigwedge_{i=1}^{n} \mathcal{E}(f_i;\theta) \in [\tilde{\mathcal{E}}_q[f_i;\theta] - \epsilon(q,\delta,N), \tilde{\mathcal{E}}_q[f_i;\theta] + \epsilon(q,\delta,N)]$ ▷ Add samples to empirical system
      $\theta_{m+1} \leftarrow$ SOLVECONSTRAINTSYSTEM($\hat{\phi}_{m+1}$)
      if $\hat{\phi}_{m+1}$ is unsatisfiable **then**
        return Unsat ▷ Original system is unsatisfiable with prob. $\geq 1-\delta$
      else $m \leftarrow m+1$
      end if
    end if
  end while
  return Unknown ▷ Reached maximal iterations I
end function

In each iteration, we perform a hypothesis test by checking the satisfiability or unsatisfiability, adding to a maximum amount of hypothesis tests of I for each of both results. As the samples we use at one iteration are used in the next iteration as well by adding the corresponding empirical estimate as an additional constraint, the tests are not independent to each other. Note that we have to use a Bonferroni correction $\delta^c \leftarrow \frac{\delta}{I}$ to compensate for this dependency (see [32]). As we would like to have the

final result to hold with confidence $1-\delta$, the Bonferroni correction requires each of the hypothesis tests (there is a maximum of I) to give a valid result with at least $1-\frac{\delta}{I}$, thereby guaranteeing that the overall probability of obtaining a valid result is bounded by $(1-\frac{\delta}{I})^I \geq 1 - I\frac{\delta}{I} = 1-\delta$.

Taken together, this implies that whenever the algorithm terminates with a definite result, this result is sound with a confidence $\geq 1-\delta$. Hence, any parameter instance generated will actually satisfy the feasibility condition Definition 1 with probability $\geq 1-\delta$. Likewise, an infeasibility result reported implies that the problem actually is infeasible with probability $\geq 1-\delta$.

5 Conclusion

Addressing the quest for automatic analysis tools covering the state dynamics of hybrid discrete-continuous systems, we have over the past decade developed a rich set of constraint solvers facilitating their symbolic or mixed symbolic-numeric analysis, starting from the first practical SAT-modulo-theory solver for real arithmetic involving transcendental functions and thus going beyond the confined decidable fragments of arithmetic (iSAT, [14]) over the seamless integration of safe ODE enclosures in SAT-modulo-theory solving (odeSAT, [11]) to stochastic extensions of SAT-modulo-theory (SSMT and CSSMT, [15, 19]). These techniques permit key-press verification of bounded safety properties of the embedded system within its physical environment, whereby both qualitative, i.e., normative, and quantitative, stochastic models of system dynamics are supported. The related tools have been developed within the Transregional Collaborative Research Action SFB-TR 14 "Automatic Verification and Analysis of Complex Systems" (AVACS, www.avacs.org) and the Research Training Group DFG-GRK 1765: "System Correctness under Adverse Conditions" (SCARE, `scare.uni-oldenburg.de`) and some of them, like the iSAT tool, are freely available from the respective web sites.

Within this chapter, we have in particular elaborated on the most recent methods and tools from that series. These are able to, first, solve quantitative bounded reachability of stochastic hybrid systems involving both discrete and continuous non-determinism and stochasticity and, second, synthesize feasible parameters for probabilistic branching in such systems satisfying multi-objective design goals w.r.t. expected cost/rewards in parametric stochastic hybrid systems. The workhorses here are CSSMT solving (Continuous Stochastic Satisfiability Modulo Theory [19]) and a novel blend of statistical model checking and arithmetic constraint solving facilitated by a symbolic version of importance sampling [18]. We expect such combinations to have a much broader area of application, as they can be used for automatically mining from samples a formal model of rigorously controlled epistemological validity: the methods provide a learning scheme yielding a formal constraint model whose validity can be guaranteed up to a quantifiable confidence, as explained in Sect. 4. We are currently trying to exploit that latter fact for porting formal verification to

safety-critical embedded software inherently devoid of a formal functional specification, like the computer vision components with their object classifiers trained by machine learning that are central to future automated driving functions.

References

1. Alur, R., Courcoubetis, C., Henzinger, T.A., Ho, P.H.: Hybrid automata: an algorithmic approach to the specification and verification of hybrid systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems. Lecture Notes in Computer Science, vol. 736, pp. 209–229. Springer, New York (1993)
2. Alur, R., Courcoubetis, C., Halbwachs, N., Henzinger, T.A., Ho, P.H., Nicollin, X., Olivero, A., Sifakis, J., Yovine, S.: The algorithmic analysis of hybrid systems. Theor. Comput. Sci. **138**, 3–34 (1995)
3. Audemard, G., Bozzano, M., Cimatti, A., Sebastiani, R.: Verifying industrial hybrid systems with MathSAT. ENTCS **89**(4) (2004)
4. Barrett, C., Sebastiani, R., Seshia, S.A., Tinelli, C.: Satisfiability modulo theories. In: Biere et al. [7], chap. 26, pp. 825–885
5. Bellman, R.: A Markovian decision process. J. Math. Mech. **6**, 679–684 (1957)
6. Biere, A., Cimatti, A., Clarke, E., Zhu, Y.: Symbolic model checking without BDDs. In: TACAS'99. Lecture Notes in Computer Science, vol. 1579, pp. 193–207. Springer, New York (1999)
7. Biere, A., Heule, M.J.H., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability. Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press, Amsterdam (2009)
8. Bousquet, O., Boucheron, S., Lugosi, G.: Introduction to statistical learning theory. Advanced Lectures on Machine Learning, pp. 169–207. Springer, New York (2004)
9. Chaochen, Z., Hoare, C.A.R., Ravn, A.P.: A calculus of durations. Inf. Process. Lett. **40**(5), 269–276 (1991)
10. Chaochen, Z., Ravn, A.P., Hansen, M.R.: An extended duration calculus for hybrid real-time systems. In: Grossman, R.L., Nerode, A., Ravn, A.P., Rischel, H. (eds.) Hybrid Systems. Lecture Notes in Computer Science, vol. 736, pp. 36–59. Springer, New York (1992)
11. Eggers, A., Fränzle, M., Herde, C.: SAT modulo ODE: a direct SAT approach to hybrid systems. In: Cha, S.S., Choi, J.Y., Kim, M., Lee, I., Viswanathan, M. (eds.) Proceedings of the 6th International Symposium on Automated Technology for Verification and Analysis (ATVA'08). Lecture Notes in Computer Science, vol. 5311, pp. 171–185. Springer, New York (2008)
12. Fränzle, M., Herde, C.: Efficient proof engines for bounded model checking of hybrid systems. In: Ninth International Workshop on Formal Methods for Industrial Critical Systems (FMICS 04), Electronic Notes in Theoretical Computer Science (ENTCS). Elsevier (2004)
13. Fränzle, M., Herde, C., Ratschan, S., Schubert, T., Teige, T.: Interval constraint solving using propositional SAT solving techniques. In: Proceedings of the CP 2006 First International Workshop on the Integration of SAT and CP Techniques, pp. 81–95 (2006)
14. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. JSAT **1**(3–4), 209–236 (2007)
15. Fränzle, M., Hermanns, H., Teige, T.: Stochastic satisfiability modulo theory: a novel technique for the analysis of probabilistic hybrid systems. In: Egerstedt, M., Mishra, B. (eds.) Proceedings of the 11th International Conference on Hybrid Systems: Computation and Control (HSCC'08). Lecture Notes in Computer Science (LNCS), vol. 4981, pp. 172–186. Springer, New York (2008)
16. Fränzle, M., Teige, T., Eggers, A.: Engineering constraint solvers for automatic analysis of probabilistic hybrid automata. J. Logic Algebr. Program. **79**, 436–466 (2010)

17. Fränzle, M., Hahn, E.M., Hermanns, H., Wolovick, N., Zhang, L.: Measurability and safety verification for stochastic hybrid systems. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control, pp. 43–52. ACM (2011)
18. Fränzle, M., Gerwinn, S., Kröger, P., Abate, A., Katoen, J.: Multi-objective parameter synthesis in probabilistic hybrid systems. In: Sankaranarayanan, S., Vicario, E. (eds.) Formal Modeling and Analysis of Timed Systems - 13th International Conference, FORMATS 2015, Madrid, Spain, 2–4 September 2015, Proceedings. Lecture Notes in Computer Science, vol. 9268, pp. 93–107. Springer, New York (2015)
19. Gao, Y., Fränzle, M.: A solving procedure for stochastic satisfiability modulo theories with continuous domain. In: Campos, J., Haverkort, B.R. (eds.) Quantitative Evaluation of Systems, 12th International Conference, QEST 2015, Madrid, Spain, 1–3 September 2015, Proceedings. Lecture Notes in Computer Science, vol. 9259, pp. 295–311. Springer, New York (2015)
20. Granvilliers, L., Benhamou, F.: Realpaver: an interval solver using constraint satisfaction techniques. ACM Trans. Math. Softw. (TOMS) **32**(1), 138–156 (2006)
21. Groote, J.F., Koorn, J.W.C., van Vlijmen, S.F.M.: The safety guaranteeing system at station Hoorn-Kersenboogerd. In: Conference on Computer Assurance, pp. 57–68. National Institute of Standards and Technology (1995)
22. Henzinger, T.A.: The theory of hybrid automata. In: Inan, M., Kurshan, R. (eds.) Verification of Digital and Hybrid Systems. NATO ASI Series F: Computer and Systems Sciences, vol. 170, pp. 265–292. Springer, New York (2000)
23. Herde, C., Eggers, A., Fränzle, M., Teige, T.: Analysis of hybrid systems using HySAT. In: The Third International Conference on Systems (ICONS 2008), pp. 196–201. IEEE Computer Society (2008)
24. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**, 13–30 (1963)
25. Julius, A.A.: Approximate abstraction of stochastic hybrid automata. In: Hespanha, J.P., Tiwari, A. (eds.) Hybrid Systems: Computation and Control: 9th International Workshop, HSCC 2006, Santa Barbara, CA, USA, 29–31 March 2006. Proceedings. Lecture Notes in Computer Science, vol. 3927, pp. 318–332. Springer, New York (2006)
26. Lee, E.A., Zheng, H.: Operational semantics of hybrid systems. In: Morari, M., Thiele, L. (eds.) HSCC'05. Lecture Notes in Computer Science, vol. 3414. Springer, New York (2005)
27. Littman, M.L., Majercik, S.M., Pitassi, T.: Stochastic boolean satisfiability. J. Autom. Reason. **27**(3), 251–296 (2001)
28. Majercik, S.M.: Stochastic boolean satisfiability. In: Biere et al. [7], chap. 27, pp. 887–925
29. Majercik, S.M., Littman, M.L.: Maxplan: a new approach to probabilistic planning. AIPS **98**, 86–93 (1998)
30. Majercik, S.M., Littman, M.L.: Contingent planning under uncertainty via stochastic satisfiability. In: AAAI/IAAI, pp. 549–556 (1999)
31. McDiarmid, C.: On the method of bounded differences. Surv. Comb. **141**(1), 148–188 (1989)
32. Miller, R.G.: Simultaneous Statistical Inference. Springer, New York (1981)
33. Papadimitriou, C.H.: Games against nature. J. Comput. Syst. Sci. **31**(2), 288–301 (1985)
34. Ravn, A.P., Rischel, H.: Requirements capture for embedded real-time systems. In: Proceedings of IMACS-MCTS'91 Symposium on Modelling and Control of Technological Systems, Villeneuve d'Ascq, France, 7–10 May, vol. 2, pp. 147–152. IMACS (1991)
35. Sproston, J.: Decidable model checking of probabilistic hybrid automata. In: Joseph, M. (ed.) Formal Techniques in Real-Time and Fault-Tolerant Systems. Lecture Notes in Computer Science, vol. 1926, pp. 31–45. Springer, New York (2000)
36. Sproston, J.: Model checking for probabilistic timed and hybrid systems. Ph.D. thesis, University of Birmingham (2001)
37. Teige, T.: Stochastic satisfiability modulo theories: a symbolic technique for the analysis of probabilistic hybrid systems. Ph.D. thesis, Universität Oldenburg (2012)
38. Teige, T., Fränzle, M.: Stochastic satisfiability modulo theories for non-linear arithmetic. Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, pp. 248–262. Springer, New York (2008)

39. Tokdar, S.T., Kass, R.E.: Importance sampling: a review. Wiley Interdiscip. Rev.: Comput. Stat. **2**(1), 54–60 (2010)
40. Tseitin, G.: On the complexity of derivations in propositional calculus. In: Studies in Constructive Mathematics and Mathematical Logics (1968)
41. Vapnik, V.N.: Statistical Learning Theory, vol. 1. Wiley, New York (1998)
42. Younes, H.L.S., Simmons, R.G.: Probabilistic verification of discrete event systems using acceptance sampling. In: Computer Aided Verification, 14th International Conference, CAV 2002, Copenhagen, Denmark, 27–31 July 2002, Proceedings, pp. 223–235 (2002)
43. Zhang, L., She, Z., Ratschan, S., Hermanns, H., Hahn, E.M.: Safety verification for probabilistic hybrid systems. In: Proceedings of the 22nd International Conference on Computer Aided Verification. Lecture Notes in Computer Science, vol. 6174, pp. 196–211. Springer, New York (2010)
44. Zhang, Y., Sankaranarayanan, S., Somenzi, F.: Statistically sound verification and optimization for complex systems. In: Cassez, F., Raskin, J.F. (eds.) Automated Technology for Verification and Analysis. Lecture Notes in Computer Science, vol. 8837, pp. 411–427. Springer, New York (2014)

http://www.springer.com/978-3-319-48627-7

Provably Correct Systems

Hinchey, M.; Bowen, J.P.; Olderog, E.-R. (Eds.)

2017, XV, 328 p. 84 illus., 48 illus. in color., Hardcover
ISBN: 978-3-319-48627-7