Int J Softw Tools Technol Transfer (2015) 17:485–504
DOI 10.1007/s10009-014-0329-y

SMC

Statistical model checking for stochastic hybrid systems involving nondeterminism over continuous domains

Christian Ellen · Sebastian Gerwinn · Martin Fränzle

Published online: 3 August 2014

Abstract Behavioral verification of technical systems involving both discrete and continuous components is a common and demanding task. The behavior of such systems can often be characterized using stochastic hybrid automata, leading to verification problems which can be formalized and solved using stochastic logic calculi such as stochastic satisfiability modulo theory (SSMT). While algorithms for discharging proof obligations in SSMT form exist, their applicability is limited due to the computational complexity, which often increases exponentially with the number of quantified variables. Recently, statistical model checking has been successfully applied to stochastic hybrid systems, thereby increasing the size of the system for which verification problems is tractable. However, being based on randomized simulation, these methods usually cannot handle non-determinism. In previous work, we have deviated from the usual approach of simulating the model and rather proposed a statistical method for SSMT solving which, being based on statistical AI planning algorithms, can also treat non-determinism over a finite domain. Here, we extend this previous work to the case of continuous domains. In particular, using ideas from noisy optimization, we adaptively build up a decision tree recording the findings and guiding further exploration, thereby favoring the currently most promising sub-domain. The non-determinism is resolved by translating the satisfaction problem into an optimization problem, thereby computing both optimistic and pessimistic bounds on the probability of satisfaction. At each stage of the evaluation process, we show how to obtain confidence statements about the probability of satisfaction for the overall SSMT formula, including reliable estimates on the optimal resolution of any non-deterministic choice involved.

Keywords Statistical model checking · Stochastic hybrid systems · Non-determinism · SSMT

C. Ellen (✉) · S. Gerwinn
Transportation, OFFIS - Institute for Information Technology, Escherweg 2, 26121 Oldenburg, Germany
e-mail: christian.ellen@offis.de

S. Gerwinn
e-mail: sebastian.gerwinn@offis.de

M. Fränzle
Hybrid Systems, Carl von Ossietzky Universiät Oldenburg - Department of Computer Science, Escherweg 2, 26121 Oldenburg, Germany
e-mail: fraenzle@informatik.uni-oldenburg.de

1 Introduction

Today's technical systems are complex and interact with each other as well as with their physical environment. Mathematically, the behavior of such ensembles can frequently be characterized using stochastic hybrid systems, where discrete controllers act within a continuous, potentially stochastic environment. Even in the absence of obvious stochastic influences, complex environments are often more accurately modeled using probabilistic abstractions, due to unknown or unobservable underlying causes of the observable behavior. Due to the heterogeneity of dynamic effects and the tight interaction of variables of different types, the analysis of such stochastic hybrid systems is notoriously difficult, calling for automated verification methods. In the recent past, formal verification of digital systems has been successfully implemented in commercial tools with wide-spread applications in industry and academia. Among the most successful verification methods for finite-state systems is bounded model checking (BMC) [1]. Although BMC was originally formulated for discrete transition systems, it can also be applied to hybrid, yet non-probabilistic systems. The BMC formulae arising from such systems comprise complex boolean com-

binations of arithmetic constraints over discrete as well as real-valued variables. Current satisfiability-modulo-theory (SMT) solvers over arithmetic theories are addressing these systems [2]. Proposed models of hybrid systems which are augmented with probabilities vary with respect to the degree of continuous dynamics, the support for random phenomena, and the degree to which they support non-determinism and compositionality [3]. Stochastic hybrid systems in their full generality, covering this variety of phenomena, have a broad range of potential applications. Especially the combination of stochastic and non-deterministic phenomena is of great interest. For example, during the early system design phase most parts of the system have not been fully defined, yet checking requirements can decrease the development cost substantially. An analysis in these early stages can identify potential problems and inconsistencies. The computation of a pessimistic estimate for the yet unexplored design space of a system can be accounted for by introducing non-deterministic components. In addition, the already defined requirements could also potentially include stochastic elements (e.g., mean times between failures).

In [4], a formalism called stochastic satisfiability modulo theory (SSMT) was introduced as a unification of stochastic propositional satisfiability [5] and satisfiability modulo theory. With the SSMT formalization and the corresponding constraint solvers [4], it is possible to model networks of hybrid automata [6] which include both discrete probabilistic branching and non-deterministic decisions on transitions.

The computational complexity to discharge the proof obligations by exact and exhaustive stochastic constraint solving, however, often is prohibitive. In [7], we therefore extended the model checking procedure for solving SSMT problems to a more scalable version based on statistical model checking (SMC) and the upper bound confidence algorithm for trees [8]. In contrast to model checking, the SMC method generates results which can be guaranteed with a certain level of confidence only, i.e. have a residual, though well-defined probability of deviating erratically.

Overcoming its limitation to finite branching both in probabilistic and non-deterministic choices, this method is extended within this paper to also support hybrid systems with continuously-valued probabilistic and non-deterministic influences, enabling SSMT to address problems from the domain of Stochastic Hybrid Systems [9] and stochastic optimal control. The core idea is to adaptively discretize the continuous domains of all quantified variables by applying a strategy derived from hierarchical optimistic optimization (HOO) [10] and to apply our previous statistical SSMT procedure to the discretization.

The paper is structured as follows: Sect. 2 provides a short survey of related work on SSMT and on statistical methods for model-checking randomized hybrid systems. In Sect. 3, we review the HOO algorithm, which forms the basis of the novel algorithm proposed in this article. Then we present our approach as an extension of the HOO algorithm to nested continuous quantification in Sect. 4. Section 5 contains an evaluation of the proposed algorithm and Sect. 6 concludes the paper.

2 Related work and background

2.1 Stochastic hybrid automata and SSMT

We are interested in satisfiability problems concerning probabilistic hybrid systems, which we illustrate with a simple example in Fig. 1. This example uses discrete non-deterministic choices of transitions (e.g., transition t_1 and

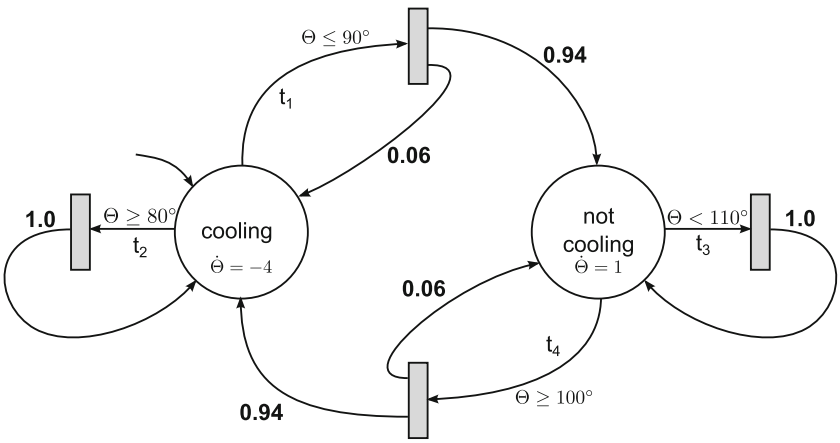


Fig. 1 Example of a probabilistic hybrid system with discrete probabilistic and non-deterministic decisions. This simple cooling system can either be in the state `cooling` or `not cooling`. Within the cooling state, the temperature Θ is decreased constantly whereas in the `not cooling` state, the temperature rises. *Bold numbers* at the edges reflect transition probabilities for the given probabilistic transitions, which can be activated once the guards (inequalities) hold. For this system, we might be interested in the probability of overheating (e.g., $\Theta > 115°$) within a given time frame

t_2 are both enabled for temperatures $\Theta \in [80°, 90°]$) and discrete probabilistic branches (e.g., taking transition t_1 will lead with a probability of 94 % to the `not cooling` state or stays in the `cooling` state with probability 6 %). This limitation to discrete numbers of choices will be addressed within the scope of this article to support a continuum of choices.

For this particular system, we might be interested in guaranteeing that the probability of overheating is lower than a given threshold. To this end, we use satisfiability modulo theory (SMT) to formalize the transition relation as an arithmetic satisfiability problem, obtaining a symbolic representation of all possible transitions in the form of a pre/post relation between values x of state variables before and x' of state variables after a transition. We denote the SMT formula which indicates the satisfiability for the given transition by $\phi(x_1, \ldots, x_n)$. The main task for the SMT solver is then to determine the existence of a satisfying variable assignment for a given SMT formula $\phi(x_1, \ldots, x_n)$. In the cooling example, this corresponds to an assignment of a temperature trajectory, given an initial state and a starting temperature.

More generally, SMT consists of a decision problem for first-order logical formulas over a given background theory (e.g., the arithmetic theories over real numbers, including multiplication as well as transcendental functions). To address the discrete non-deterministic and probabilistic choices we use Stochastic SMT (SSMT) as an extension of SMT [11]. An SSMT formula Φ extends an SMT formula ϕ by adding a prefix of quantified variables $Q_1 X_i, \ldots, Q_n X_n$. Every quantifier Q_i of the prefix binds one variable X_i of ϕ and is either randomized ($\bowtie_{X_i}$), existential ($\exists_{X_i}$), or universal ($\forall_{X_i}$). Every quantified variable X_i in the original definition has a finite domain $\mathcal{X}_i$. In the randomized case, every value $x_j \in \mathcal{X}_i$ is associated with a probability $P(X_i = x_j)$, modeling the likelihood that the corresponding transition is chosen. The other quantifier types model different ways to resolve non-determinism: $\exists$ by maximizing the probability of satisfying the remaining formula over all domain values and analogously $\forall$ by minimizing the probability. The conditional probability distribution $P(\cdot|\cdot)$ of a randomized variable X_i can be conditioned on choices of values for variables $X_0, \ldots, X_{i-1}$ which appear left of X_i in the prefix. This allows to model dependencies between random variables as well as, for example, parametric distributions where the parameters depend on other quantified variables of the prefix. In addition, the presented Definition 1 is already extended towards supporting infinite domains. In this case, $p(\cdot|\cdot)$ denotes the probability density function for randomized quantifiers.

Definition 1 The semantics of an SSMT formula Φ is recursively defined over its probability of satisfaction $P(\Phi)$ using Q for the remainder for the quantifier prefix (cf. [6]):

$$
\begin{aligned}
&1.\ P(\epsilon : \phi) && = 0 \text{ if } \phi \text{ is unsatisfiable.}\\
&2.\ P(\epsilon : \phi) && = 1 \text{ if } \phi \text{ is satisfiable.}\\
&3.\ P(\exists_{X_i} Q : \phi) && = \max_{x \in \mathcal{X}_i} P\,(Q : \phi[X_i = x])\,.\\
&4.\ P(\forall_{X_i} Q : \phi) && = \min_{x \in \mathcal{X}_i} P\,(Q : \phi[X_i = x])\,.\\
&5.\ P(\bowtie_{X_i} Q : \phi) && = \begin{cases} \sum_{x \in \mathcal{X}_i} P(X_i = x | X_{\setminus i}) \cdot \\ \quad P\,(Q : \phi[X_i = x]) & \text{if } |\mathcal{X}_i| < \infty \\ \int_{\mathcal{X}_i} p(X_i = x | X_{\setminus i}) \cdot \\ \quad P\,(Q : \phi[X_i = x])\,\mathrm{d}x & \text{else.} \end{cases}
\end{aligned}
\tag{1}
$$

Here, we used the shorthand notation $X_{\setminus i}$ within the case of a randomized quantifier to represent the set of all variables within the quantifier prefix which appear before the ith one.

Using Definition 1, the cooling system S can be encoded as a SSMT formula which can be used in a bounded model checking procedure for a given maximum depth k. A complete definition of the construction as well as an evaluation of a network of hybrid automata can be found in [6]. In a nutshell, the initial state of the automaton is translated into a predicate $INIT_S(0)$ which ensures that in the beginning the initial state of the automaton is active (here: $INIT_S(0)$:=`cooling`) and all occurring variables are restricted to their corresponding initial values or ranges. The transitions of the hybrid automaton are translated into predicates $TRANS_S(j-1, j)$ which encode a transition relation from step $j-1$ to the depth j for each $1 < j < k$.

The non-deterministic selection of any of the enabled transitions is modeled by introducing a new existential quantified variable tr for each step which resolves the non-deterministic selection by maximizing the probability of reaching the target state (e.g., for Fig. 1: $tr \in \{t_1, \ldots, t_4\}$). For the transitions a randomized variable named pc is introduced which models the probabilistic choices of target states. The probabilities of pc are conditioned to the transition selected by tr. This encoding scheme also respects the hybrid nature of the automaton by not introducing a time discretization, only the maximum number of transition steps (jumps) is limited to k by unrolling, not the time spent within a location (flow). Nevertheless, the time has to be bounded by the domain of its variable in the formula. Finally, a predicate $TARGET(j-1)$ is added for each step which indicates whether the desired target state of the system has been reached:

$$
\begin{aligned}
&BMC_{S,\text{target}}(k) := INIT_S(0)\\
&\wedge \bigwedge_{j=1}^{k} \begin{pmatrix} (\neg TARGET_S(j-1) \implies TRANS_S(j-1, j)) \\ \wedge (TARGET_S(j-1) \implies (x_{j-1} \equiv x_j)) \end{pmatrix}\\
&\wedge TARGET(k)
\end{aligned}
\tag{2}
$$

All quantified variables are replicated for each step and are added in the prefix of $BMC_{S,\text{target}}(k)$.

From Definition 1, we also see that the resulting satisfiability problem can be written as a nested optimization–expectation problem. For example, if we set $k = 1$ and encode the cooling example, the resulting SSMT formula consists of a single existential quantifier followed by a randomized one and the satisfiability problem can be written as follows:

$$\begin{aligned} P(\exists_{tr} \biguplus_{pc|tr} : \phi) &= \max_{t \in tr} \mathbb{E}_{pc|t}\left[P(\phi)\right] \\ &= \max_{t \in tr} \left(\sum_{c \in pc} P(\phi(t,c)) P(c|t) \right) \\ &\quad \text{with: } \phi = BMC_{S,\text{target}}(1) \end{aligned} \tag{3}$$

This would correspond to first selecting a transition from the `cooling` state and then choosing one of the transition targets at random, according to the probabilities. The existential quantifier corresponds to a pessimistic choice of transitions in terms of consequences with respect to overheating.

Equation (3) suggests we should approximate the expectation via a sampling-based scheme, if the set of pc is too large. If we use the average of samples generated according to $P(c|t)$, we observe only a noisy estimate $\hat{\mathbb{E}}_{c|t}$ of the true underlying function $\mathbb{E}_{c|t}$. Using confidence intervals for this estimation, we obtain an uncertainty estimate for the expectation, *i.e.,* randomized quantifier which has then to be propagated through other quantifiers to obtain an overall uncertainty estimate on Φ. Computationally, we are interested in an efficient way of calculating Eq. (3), that is to efficiently search for promising t choices to evaluate.

A common representation of an SSMT formula uses a decision tree for the variables in the quantifier prefix, where the nodes are the variables (in order of their occurrence in the prefix) and the decisions are the domain values. The leafs of the tree are replications of ϕ, where every quantified variable is substituted according to the path in the decision tree. Therefore, every leaf represents an SMT problem of its own. As solving these SMT problems at the leaves of the decision tree is a time-consuming problem, most existing work focuses on minimizing the number of evaluations of the leaves by carrying information from one leaf evaluation to another using conflict learning.

By extending SSMT with quantifiers with continuous domains, SSMT can be used to model purely continuous stochastic systems as well as extended probabilistic hybrid automata which can use a mixture of discrete quantifiers (e.g., for the selection of transitions) and continuous quantifiers (e.g., for the dynamics while staying within a state). In particular such continuous (random) quantifiers arise, if sensor measurements are accounted for in the verification of a controller interacting with the physical environment via noisy sensors (see [12]).

2.2 Model checking of stochastic hybrid automata: related work

In the following, we briefly review statistical model checking and the existing work on SSMT solving.

Statistical model checking. Inspired by classical hypothesis testing based on a set of data samples, statistical model checking uses generated traces of the system under investigation to estimate the probability with which a given property holds. To this end, the system has to provide a trace generator [13,14]. As such a trace generator can only generate finite trajectories, only bounded time properties can be checked with such an approach. However, if more structure of the underlying model is known (e.g., a continuous time Markov chain [15]) these structures may be exploited to reason about unbounded properties as well. Also, additional information, for example, in terms of associated costs, can be incorporated [16]. Instead of hypothesis testing, one can also impose a prior distribution on the probability of satisfaction and update a Bayesian belief sequentially as new samples are drawn from the generative model [17]. Most approaches are only applicable, if no decisions are needed to be resolved other than random nondeterministic ones, however, see [18] for a recent extension to Markov Decision Processes.

SSMT solving. Based on the iSAT [19] algorithm for SMT problems, an algorithm called SiSAT [6] has been developed to solve SSMT problems efficiently. It implements a fully symbolic solving procedure based on the traversal of the prefix tree, using extended conflict driven clause learning (CDCL) procedures and pruning rules. The computed probability of satisfaction comes with absolute certainty, that is SiSAT terminates, if the probability is guaranteed to be larger than a given threshold or it has been computed exactly. The pruning rules allow SiSAT to ignore parts of the quantifier tree if the outcome of the decisions could be inferred or has no impact on the result (e.g., if the target threshold has already been exceeded or cannot be reached anymore). Otherwise, the algorithm has to perform an exhaustive search over the state space. Due to this exhaustive search, the number of leaves in the tree—and hence the number of SMT problems to solve—depends exponentially on the number of quantified variables in the prefix. Although SiSAT has to examine exponentially many leaves in the worst case, the memory usage is still limited, as the tree is searched in a depth-first manner.

3 Review of hierarchical optimistic optimization (HOO) algorithm

Statistical solving of a nested SSMT formula [see Eq. (1)] is a case of maximizing an expected value based only on noisy

function evaluations, like those provided by Monte Carlo sampling. This noisy optimization problem has recently been studied in [10], resulting in an algorithm called hierarchical optimistic optimization (HOO). In the following, we first review this algorithm, as it forms the basis of our approach to solve problems involving nested, alternating quantifiers. We then demonstrate how this algorithm can be adapted to solve a slightly modified, more general parametric case, which in turn can be used to solve the nested, alternating quantification. Mathematically, the special case considered by the HOO algorithm can be written as follows:

$$\max_{x\in\mathcal{X}} \mu(x) = \max_{x\in\mathcal{X}} \int_{\mathcal{Y}} f(x,y)\,p(y|x)\mathrm{d}y \tag{4}$$

It is thereby assumed that the conditional probability density function p and the function f are unknown[1], but that one can draw samples y for a given x and therefore of the random variable z which results from using this sample and plug it into the function f. In that way, one could approximate the mean function μ at any point x via the empirical average, obtaining a noisy estimate. Furthermore, there are a few regularity assumptions on the function μ, which are necessary to solve the optimization problem. For the sake of simplicity, we restate the main assumptions in a more restrictive way compared to [10]:

1. Sample boundedness: it is assumed, that each sample and hence the mean-payoff function is bounded, i.e., $P(z \in [0,1]) = 1$ and therefore $\mu(x) \in [0,1]$.
2. Domain boundedness: it is assumed that the domain $\mathcal{X}$ is bounded, *i.e.*, $\mathcal{X} \subseteq [a,b]^n$ with $|a|, |b|, n < \infty$.
3. Lipschitz continuity: The function μ is assumed to be locally (around its maximum) Lipschitz continuous with known constant ν.

These assumptions can be weakened considerably by assuming only weakly Lipschitz continuous functions in a topological space, where the Lipschitz continuity holds with respect to a similarity measure. However, for the sake of simplicity, we formulated the above, more restrictive assumptions in $\mathbb{R}^n$.

The HOO algorithm successively acquires knowledge about the noisy function to be optimized. More precisely, the function is more accurately explored around its apparent maximum and is investigated with less precision anywhere else. To this end, the optimization domain is decomposed into a binary tree, which is explored using an upper confidence bound (UCB) strategy [8,20]. Within the HOO algorithm, the domain $\mathcal{X}$ is represented by a covering tree $\mathcal{X}_{h,i}$:

$$\mathcal{X}_{0,1} = \mathcal{X},$$
$$\mathcal{X}_{h,i} = \mathcal{X}_{h+1,2i-1} \cup \mathcal{X}_{h+1,2i}, \quad 0 \le h, \quad 1 \le i \le 2^h$$

Within this covering tree, each node h, i maintains the following quantities on an "as needed" basis:

$B_{h,i}$ Main quantity of interest as it represents the upper confidence bound for the maximal mean-payoff function value within the sub-domain $\mathcal{X}_{h,i}$ covered by a node i at depth h. If the node has not been visited yet, its B value is initialized by ∞. The B values constitute upper confidence bounds with confidence level δ, i.e., satisfy

$$P\left(\max_{x\in\mathcal{X}_{h,i}} \mu(x) < B_{h,i}\right) \ge 1-\delta$$

$T_{h,i}$ Number of times the node has been visited by the algorithm.

$U_{h,i}$ Another, yet more conservative, upper confidence bound for the maximal expected value within the covered region.

$\hat{\mu}_{h,i}$ Empirical average of the payoff within the region $\mathcal{X}_{h,i}$.

We only review the most important steps of the algorithm and refer to [10] for details. Given a cover tree within the nth iteration with its associated B values, a node is selected using an upper-confidence-bound strategy. This amounts to selecting nodes from the root to the leaves by always choosing the child with the higher B value until an hitherto unvisited node has been reached. For this node, both children within the covering tree are initialized and for each of them, an x is chosen arbitrarily from the corresponding sub-domain. A single sample is drawn for this x and the corresponding reward z in turn is used to update all U and B values on the prefix path leading to the selected sub-domain. The updates are as follows:

$$T_{h,i} \leftarrow T_{h,i} + 1 \tag{5a}$$
$$\hat{\mu}_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}_{h,i} + z/T_{h,i} \tag{5b}$$
$$U_{h,i} \leftarrow \hat{\mu}_{h,i} + \sqrt{2\log(n)/T_{h,i}} + \nu\rho^h \tag{5c}$$
$$B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\} \tag{5d}$$

In Eq. (5c), the value $\nu\rho^h$ represents the width of the maximal range of the mean function's value within the sub-domain at depth h, as ρ^h is the diameter of the sub-domain at that depth and ν represents the Lipschitz constant. The term $\sqrt{2\log(n)/T_{h,i}}$ in Eq. (5c) represents the additional slack one has to add to render the resulting U value an upper bound for the mean function with a confidence of at least $1 - n^{-4}$,

[1] In fact, the distribution is not required to allow a representation with a density function. However, as we will assume the existence of such a function later, we assume it here for the sake of simplicity.

where n is the overall number of iterations. This follows by applying Azuma's inequality for Martingales (see [10]).

Using this selection strategy together with the above-mentioned updates, the HOO algorithm iteratively explores the domain. It is important to note that each node and hence each sub-domain maintains an upper bound (together with an associated confidence term, depending on the number of times the sub-domain has been explored) on the maximum value of the mean function within its respective sub-domain. Observe that the U values are only based on the samples which are drawn at locations falling in the corresponding sub-domain, whereas the B values hierarchically combine the upper bounds from lower levels to obtain valid confidence bounds on higher (larger sub-domains) levels and therefore can only be tighter than the corresponding U values.

3.1 Example

To illustrate the HOO algorithm we use the following simple example:

$$\max_{x\in[-5,5]} \mathbb{E}_{y|x}[\mathbb{1}_C(x,y)] = \max_{x\in[-5,5]} \int \mathbb{1}_C(x,y)\mathcal{N}_y(\mu = x, \sigma^2 = 20 + x^2)\mathrm{d}y,$$
$$\text{where} \quad C = \{x, y : x^2 + y^2 \leq 10\} \tag{6}$$

The problem setup formally specified in Eq. (6) is illustrated in Fig. 2. The upper part plots the mean function, which is to be optimized and can be calculated analytically in this special case. The two-dimensional integrand of Eq. (6) is shown in the lower panel, only showing values larger than 0, i.e., $x^2 + y^2 \leq 10$.

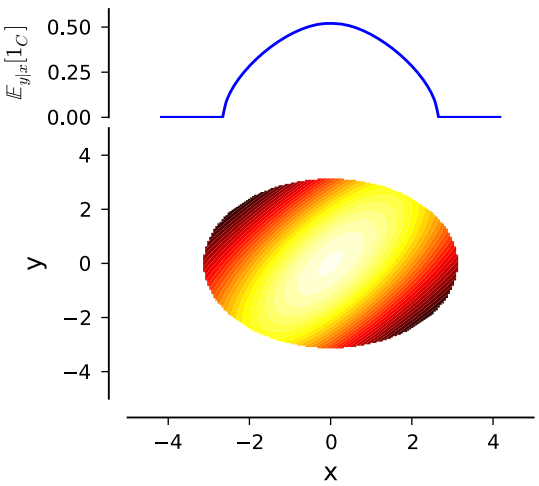


Fig. 2 Graphical illustration of the simple optimization problem in Eq. (6). The *lower part* shows the conditional Gaussian density, where only the valid region C is shown whereas the *upper part* plots the mean function to be optimized

The algorithm is applied using $\rho = 0.5$ and $\nu = 10$ as parameters and starts with a tree consisting of a single node $\mathcal{X}_{0,1}$ covering the full domain of x. The bound of its child nodes $\mathcal{X}_{1,1}$ and $\mathcal{X}_{1,2}$ (and all new nodes afterwards) is assumed to be initialized with $B_{1,1} = B_{1,2} = \infty$ (see Fig. 3, entry $n = 0$). The iteration starts with the selection of a path which follows the highest B values (UCB). Since both of the children have the same highest bound, a Bernoulli trial with $p = 0.5$ is used as a tie-breaker rule and $\pi = (\mathcal{X}_{0,1}, \mathcal{X}_{1,1})$ is the selected sampling path. The new node $\mathcal{X}_{1,1}$ covers the interval $[-5, 0]$. In a second step, a sample y value is drawn for $\mathcal{X}_{1,1}$ by setting $x = -2.5$. In case of the example in Fig. 3 the sample $y = 0.45$ results in the reward being $z = 1$. The nodes in π are updated using Eq. 5a and 5b. Then the U value for each node of the tree—not only the nodes of π—are updated using 5c. The last step is the update of the B values. Therefore, the full tree has to be iterated from the leafs to the root s.t. 5c is using the current B values of the child nodes. Figure 3 illustrates the evolution of the cover tree within the first 4 iterations of these steps.

4 Proposed method for solving nested SSMT problems

4.1 HOO algorithm using interval arithmetics

Instead of applying the HOO algorithm to the noisy optimization problem of Eq. (4), we can also use interval arithmetics in the following way to obtain a valid upper bound on the maximization problem. If we assume that we know the symbolic form of the integrand which is suitable for handling it with interval arithmetics, we can calculate the interval containing all potential values the integrand can assume. By multiplying this interval with the volume of the domain, we obtain an interval which gives a safe over-approximation of the value of the integral and hence also contains the optimum. More precisely, denoting by

$$[l, u] = \mathcal{I}_{fp} \supset \{f(x, y)p(y|x) | x \in \mathcal{X}, y \in \mathcal{Y}\}$$

an interval containing all values of the integrand, which can be determined by interval arithmetics from the symbolic form of f and p, we directly obtain the following interval bound on the maximum:

$$\max_{x\in\mathcal{X}} \mu(x) \in \underbrace{[|\mathcal{Y}| \cdot l, |\mathcal{Y}| \cdot u]}_{\mathcal{I}_{fp}\cdot|\mathcal{Y}|} \tag{7}$$

However, this bound is not very tight, as maximal and minimal values are not very tight enclosures of the integrand. The HOO algorithm, as presented in the previous section, uses the Lipschitz constant of the mean function to obtain a valid bound on the values across a range of possible x

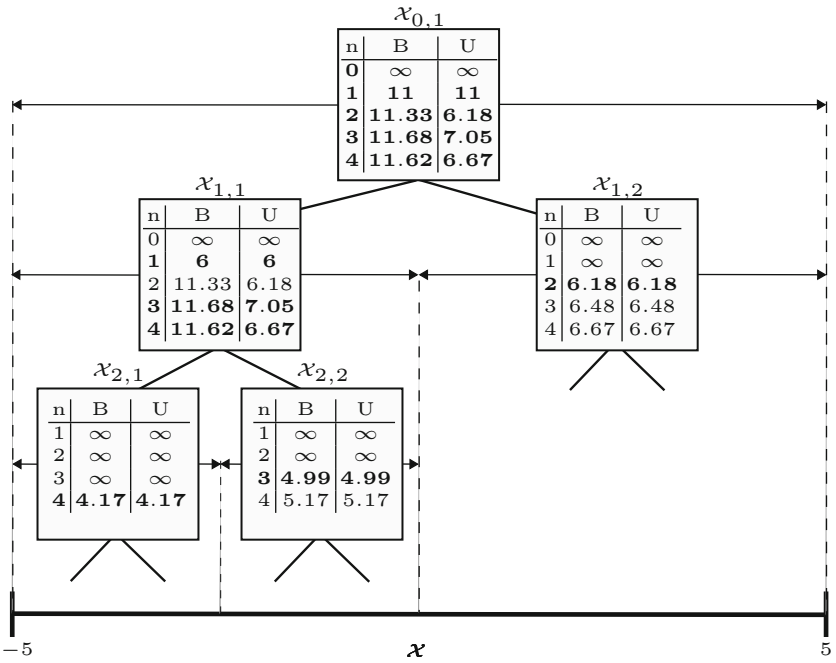


Fig. 3 Illustration of the cover tree generated by the HOO algorithm after $n = 4$ steps. Numbers marked in *bold letters* are on the sample path for that n

values by estimating the integral at a single point x statistically and bounding the variation using the Lipschitz constant. As the Lipschitz constant of the mean function in general is not known and also might be difficult to compute, even if the symbolic form of the integrand is known, we use the following combination to obtain a valid statistical bound on the maximal mean function across a given sub-region. First, we rewrite the integral in (4) by introducing a proposal distribution q following a well-known importance sampling approach:

$$\max_{x\in\mathcal{X}} \int f(x,y)p(y|x)\mathrm{d}y = \max_{x\in\mathcal{X}} \int f(x,y)\frac{p(y|x)}{q(y)}q(y)\mathrm{d}y \tag{8}$$

Here q is an arbitrary choice of a proposal distribution, which we use to sample y independent of $x \in \mathcal{X}$ as the specific x value would be necessary to sample from the original distributions characterized by $p(y|x)$. Instead of using interval arithmetics to obtain one interval for all values of x and y, we are now using it to calculate intervals for individual y_i samples generated from the proposal q:

$$\widehat{\mu(\mathcal{X})} := \frac{1}{N}\sum_{i=1}^{N}\mathcal{I}_i \tag{9}$$

$$\mathcal{I}_i \supseteq \left\{ \frac{f(x,y_i)p(y_i|x)}{q(y_i)} \middle| x \in \mathcal{X} \right\}, \qquad y_i \sim q \tag{10}$$

We thereby obtain an interval estimate containing with a certain probability all values reached by the mean function. To quantify the quality of this estimate, we can—analogously to the HOO algorithm—use the Hoeffding–Azuma inequality for Martingales:

$$\max_{x\in\mathcal{X}} \int \frac{f(x,y)p(y|x)}{q(y)}q(y)\mathrm{d}y \tag{11}$$

$$\leq \int \left(\max_{x\in\mathcal{X}} \frac{f(x,y)p(y|x)}{q(y)}\right) q(y)\mathrm{d}y =: \mu^* \tag{12}$$

$$\Rightarrow P\left(\mu^* \leq \widehat{\mu(\mathcal{X})}^+ + |\mathcal{I}_{fpq}|\sqrt{\frac{2\log\left(\delta^{-1}\right)}{N}}\right) \geq 1-\delta, \tag{13}$$

$$\mathcal{I}_{fpq} \supset \left\{ \frac{f(x,y)p(y|x)}{q(y)} | x \in \mathcal{X}, y \in \mathcal{Y} \right\} \tag{14}$$

where we used I^+ to denote the upper boundary of an interval $I = [I^-, I^+]$. Note that, instead of assuming a $[0, 1]$ support for the individual samples, we have to account for the larger variation. To this end, we have to know the support of the importance factor. $\mathcal{I}_{fpq}$ is an over-approximation of the potential values the integrand can assume, which again can be obtained using interval arithmetics. As an analogous statement holds for the minimum over all x values, we have:

$$P\left(\widehat{\mu(\mathcal{X})} + |\mathcal{I}_{fpq}|\sqrt{\frac{2\log\left(\delta^{-1}\right)}{N}}[-1,1] \supset \mu(\mathcal{X})\right) \geq 1-\delta \tag{15}$$

That is, instead of using the update in Eq. (5c), thereby accounting for the correction $(\nu\rho^h)$ obtained from the Lipschitz constant, we directly bound the variation across the

(sub-)range of x values using interval arithmetics. As a consequence of using intervals, we are not only able to keep track of the upper bound but also of the lower bound. Taken together, we compute the following updates in our interval-based HOO algorithm:

$$T_{h,i} \leftarrow T_{h,i} + 1 \tag{16a}$$

$$y_k \sim q \tag{16b}$$

$$\mathcal{I}_{k,(h,i)} = [\mathcal{I}^-_{k,(h,i)}, \mathcal{I}^+_{k,(h,i)}] \supseteq \left\{ \frac{f(x, y_k)p(y_k|x)}{q(y_k)} \middle| x \in \mathcal{P}_{h,i} \right\} \text{ using interval arithmetics} \tag{16c}$$

$$\hat{\mu}^+_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}^+_{h,i} + \mathcal{I}^+_{k,(h,i)}/T_{h,i} \tag{16d}$$

$$\hat{\mu}^-_{h,i} \leftarrow (1 - 1/T_{h,i})\hat{\mu}^-_{h,i} + \mathcal{I}^-_{k,(h,i)}/T_{h,i} \tag{16e}$$

$$U_{h,i} \leftarrow \left[\hat{\mu}^-_{h,i} - |\mathcal{I}_{fpq}|\sqrt{2\log(n)/T_{h,i}},\ \hat{\mu}^+_{h,i} + |\mathcal{I}_{fpq}|\sqrt{2\log(n)/T_{h,i}}\right] \tag{16f}$$

$$B_{h,i} \leftarrow \min\{U_{h,i}, \max\{B_{h+1,2i-1}, B_{h+1,2i}\}\} \tag{16g}$$

As the only modification compared to the HOO algorithm lies in the use of interval arithmetics to bound the variation within a given sub-region instead of applying a correction due to the Lipschitz constant, we obtain an algorithm which similarly estimates the maximum of an expectation. As interval arithmetics are guaranteed to give a valid over-approximation of the interval containing all function values, we have a guaranteed confidence bound on the maximal function value using the B values at the root of the cover tree without assuming any value for the Lipschitz constant. However, we have to assume that we know the symbolic form of the involved functions which is a much more restrictive assumption than just knowing the Lipschitz constant. Nevertheless, the evaluation can be more efficient compared to the Lipschitz-based HOO algorithm when the assumed Lipschitz constant is much larger than the actual Lipschitz constant. To illustrate the effect on the efficiency of the accuracy with which the Lipschitz constant and hence the maximal expected value is bounded, we use the Gaussian example (see Sect. 5.2). We run the HOO algorithm using the interval-based evaluation and artificially add a Lipschitz-based term to the evaluation of the samples to Eq. (16f). We added $L \cdot \text{diam}(\mathcal{P}_{h,i})$, where L represents the over-estimate of the Lipschitz constant resulting in an over-estimate of the local U-estimate. The resulting effects on the accuracy on the estimate for the original HOO algorithm are shown in Fig. 4 for a range of different L values and a fixed number of iterations.

As can be seen, over-estimating the Lipschitz constant only slightly leads to quite dramatic effects on the accuracy for a fixed number of evaluations. Nevertheless, even if the Lipschitz constant is over-estimated, the algorithm can still be shown to perform well compared to other algorithms (see [10]). Therefore, using interval arithmetics instead of a Lipschitz constant can lead to an improved efficiency without loosing any guarantees. Note, however, that also using interval arithmetic can lead to an over-estimation of the possible variation of the mean function within a sub-domain, as it only gives an over-approximation of the true image of the mean function [c.f. Eq. (10)]. Furthermore, using importance sampling can decrease the performance of the algorithm as the variation of $f\frac{p}{q}$ has to be evaluated instead of just f, reflecting the fact that the choice of a proposal distribution can have significant impact on the accuracy of the estimator. As we assume the domains to be bounded, we choose for the rest of study as proposal distribution the uniform distribution

$$q(y) := \frac{1}{|\mathcal{Y}|}. \tag{17}$$

This has the additional advantage that the number of occurrences of the variable y is reduced in the term defining the integrand, facilitating the calculation of $\mathcal{I}_k$, (h, i) and $\mathcal{I}_{fpq}$. The effect of over-approximation due to the use of interval arithmetic is reduced in turn.

The interval version of the HOO algorithm presented can also be applied if there is an additional variable entering the integrand in Eq. (8) which has an unspecified domain. To determine the value of such a parametric optimization problem, we again handle it as a problem of interval arithmetic. That is, we calculate the maximal and minimal value in the same fashion as we calculated the maximal and minimal value of the mean function as a function of the optimization argument:

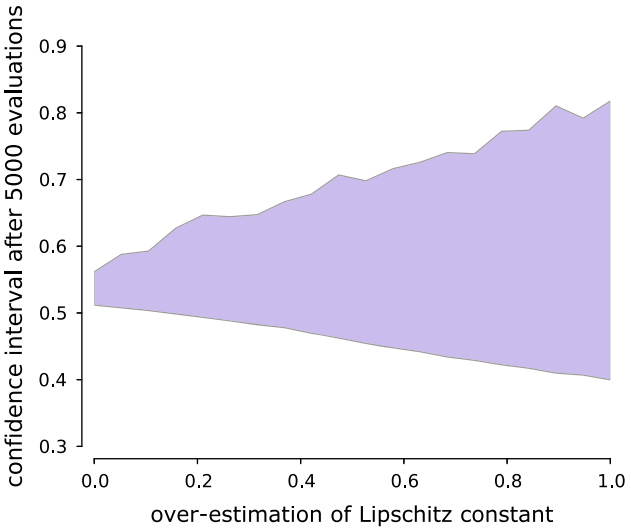


Fig. 4 Effect of the over-estimation of the Lipschitz constant on the estimated confidence interval after a fixed number (5,000) of iterations of the algorithm. The results are averaged across 10 repetitions. For evolution as a function of evaluations, see Fig. 10

$$\begin{aligned}
&\int \min_{z\in\mathcal{Z}} \min_{x\in\mathcal{X}} f(x,y,z)\frac{p(y|x,z)}{q(y)}q(y)\mathrm{d}y \\
&\leq \min_{z\in\mathcal{Z}} \min_{x\in\mathcal{X}} \int f(x,y,z)\frac{p(y|x,z)}{q(y)}q(y)\mathrm{d}y \\
&\leq \max_{z\in\mathcal{Z}} \max_{x\in\mathcal{X}} \int f(x,y,z)\frac{p(y|x,z)}{q(y)}q(y)\mathrm{d}y \\
&\leq \int f(x,y,z) \max_{z\in\mathcal{Z}} \max_{x\in\mathcal{X}} \frac{p(y|x,z)}{q(y)}q(y)\mathrm{d}y \qquad (18)
\end{aligned}$$

If we apply interval arithmetic in Eq. (10) not only to the sub-domain of $\mathcal{X}$ but also to the domain $\mathcal{Z}$, we consequently obtain lower and upper confidence bounds on the probability of satisfaction, which hold uniformly across the domain $\mathcal{Z}$.

4.2 Extension to nested problems

The case of a parametric noisy optimization problem considered in the previous section corresponds to a special parametric SSMT problem and can be generalized to arbitrarily nested quantified variables as explained in the following. Intuitively, instead of having a single cover tree representing the domain of the optimization variable (universally or existentially quantified), we represent the domain of each variable including the randomized ones with a cover tree, which is conditioned on the prefix of sub-regions of previous variables within the quantification prefix[2]. This prefix to a cover tree then acts as a parametrization in analogy to the (potentially multi-dimensional) z variable in the previous section. If a variable is ranging over a continuous domain, we use the same scheme as before, i.e., refinement of the cover tree is done during the iteration of the algorithm. If the variable has valuations in a discrete domain, we represent its domain as a tree with as many branches as there are valuations for this variable without any need for further refinement.

In Figs. 5 and 6, we illustrate the representation for the different domains and how they are iteratively refined. The path to be selected for further exploration within this forest is chosen based on an assessment of how promising this path seems to be. The details of the selection strategy are described next. Once a path is selected, one variable along the path is selected to split its domain (so far depending only on the width of the smallest sub-domain along the path). For a brief overview of the overall algorithm, see Algorithm 1.

4.2.1 Representation and assumptions

To describe the details of the algorithm, we start with the overall representation of the current state of an evaluation. The representation closely follows the procedure from the interval-based HOO algorithm from the previous section.

[2] Those variables which are mentioned within the quantifier prefix before the current variable.

Algorithm 1 Nested HOO for SSMT overview

```
function SSMT-HOO(Formula, threshold, confidence, N)
    tree ← BuildDecisionTree(Formula)
    confidence ← 1-(1-confidence)/N          ▷ Correction for sequential test
    n ← 0
    while getConfidenceBounds(tree, confidence) ≥ threshold and n ≤ N do
        path ← selectSplitPath(tree)
        sample ← sample(path)
        tree ← update(tree, sample)
        n ← n + 1
    end while
    return getConfidenceBounds(tree, confidence)   ▷ Bounds on the probability of satisfaction
end function
```

That is, it also represents the domains of the different quantifiers by a cover tree. As the quantifiers are nested, so are the cover trees, rendering the overall representation into a forest of cover trees. The leaves of a cover tree of an intermediate quantifier have a single child, which is the root of the next cover tree. Such an intermediate cover tree, however, is conditioned on the sub-domain of the entire prefix in the cover tree (see Fig. 5 for an example of an initial cover tree and Fig. 6 for an updated version). Analogous to the HOO case, quantifiers having associated continuous variables are initialized with just a single node (representing the whole domain), which is refined iteratively during the evaluation. Cover trees for quantifiers over discrete domains, however, are initialized having as many branches as there are possible valuations for the associated variable. Each node memorizes quantities reflecting the number of visits, empirical averages, and confidence bounds [see Eq. (5)]. As we are using intervals for the representations (see previous section), these quantities are memorized for both upper and lower bounds. In addition to the confidence-related quantities, the support for the integrand (including the importance weights) is memorized within each node. This is necessary as due to using the importance sampling scheme, the support of the integrand is not always limited to [0, 1], but can assume larger intervals [see Eq. (15)].

More precisely, to each quantifier Q_j within an SSMT formula $Q_1, \ldots, Q_n[\phi(x_1, \ldots, x_n)]$ we associate a cover tree $\mathcal{P}^j_{h,i}$ covering its domain. Using this cover forest, the operation associated with the quantifier Q_j (such as maximization for existential, minimization for universal, and expectation for randomized quantifier) is cast into a parametric one. That is, given a prefix for an intermediate cover tree, the result of an evaluation of the sub-formula $Q_j Q_{j+1} \ldots Q_n \phi(x_1, \ldots, x_n)$ has to consider all values for the variables within the prefix, i.e., an evaluation result has to hold uniformly across the prefix domain. We denote the prefix of variables for variable j with $x_{\searrow j} := (x_1, \ldots, x_{j-1})$. In addition, we need to associate the domain prefix with a point

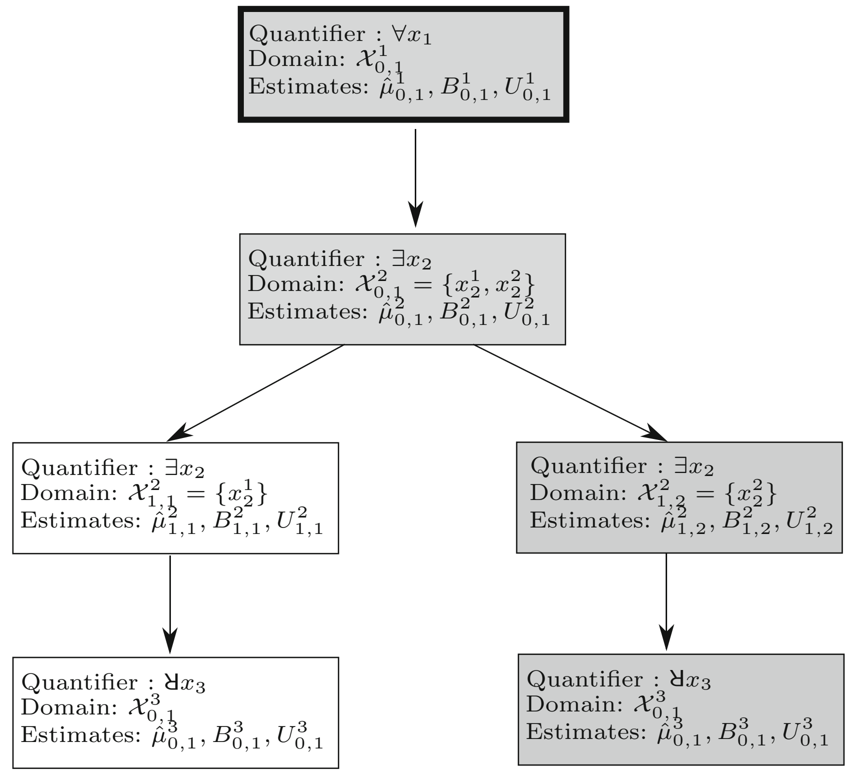


Fig. 5 Initial cover tree representation. In this example the SSMT formula is set to $\forall x_1 \exists x_2 \text{R} x_3 : \phi(x_1, x_2, x_3)$. The *highlighted path* within this tree marks the path which is selected by the selection operation (see Sect. 4.2.2). Within this path a node (in this case the root node) is selected for splitting the domain

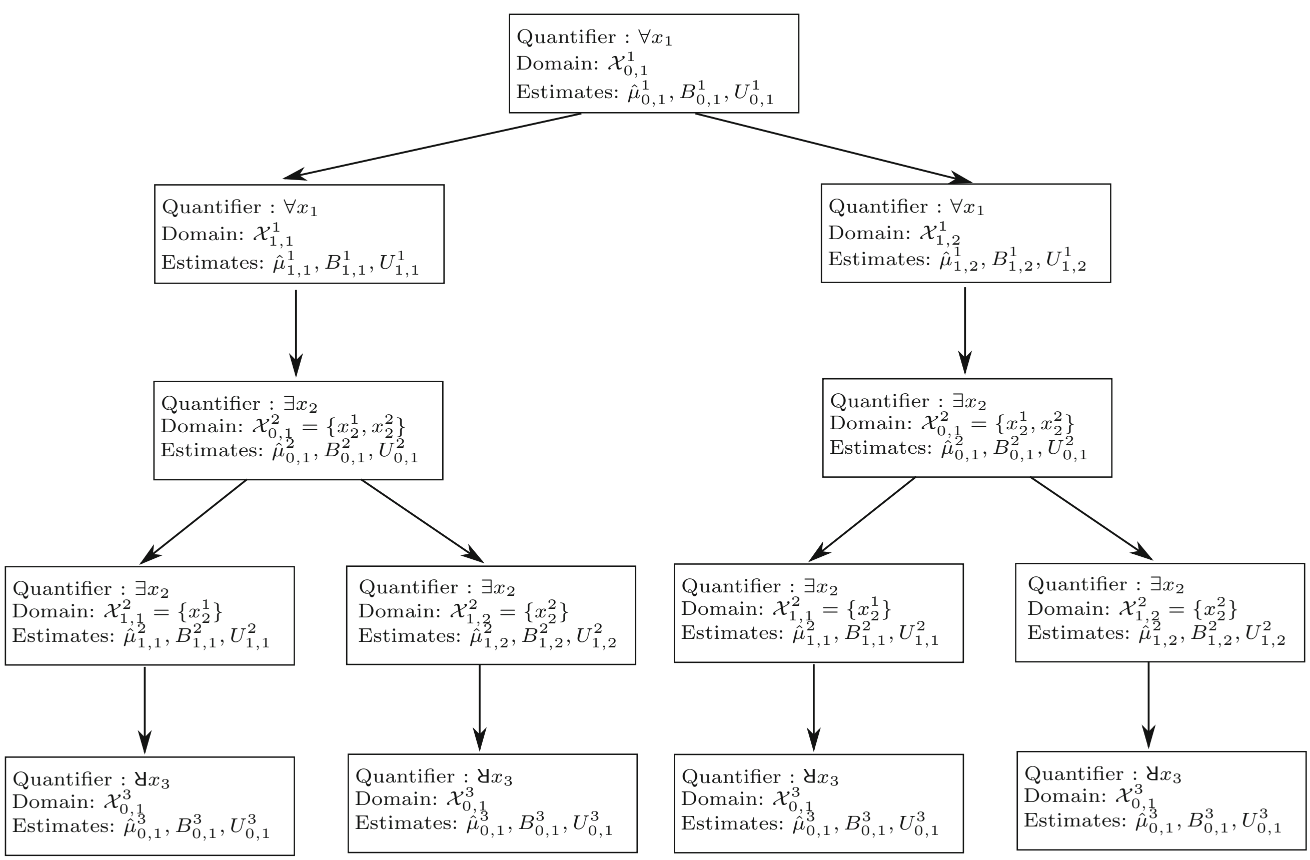


Fig. 6 Cover tree representation after a iteration. Here the updated version of this representation is shown

within the forest of cover trees. We denote this prefix with $\mathcal{P}_{\searrow j} := \times_{k=1}^{j} \mathcal{P}^{k}_{h^*(k),j^*(k)}$, where the index $(h^*(k), i^*(k))$ denotes the largest index within the cover tree of variable k on the selected path to j. For the sake of simplicity, we omitted the dependence of this prefix domain on the specific cover tree of variable j to which it is leading, as we are always concerned with complete paths within the tree. For example, in Fig. 5, considering the selected path $\mathcal{P}^1_{0,1}, \mathcal{P}^2_{0,1}, \mathcal{P}^2_{1,2}, \mathcal{P}^3_{0,1}$, the prefix $\mathcal{P}_{\searrow 3}$ would consist of just $\mathcal{P}^1_{0,1} \times \mathcal{P}^2_{1,2}$, that is $(h^*(0), i^*(1)) = (0, 1)$; $(h^*(1), i^*(2)) = (1, 2)$.

Given this notation, to solve the parametric form of the SSMT problem holding uniformly across the prefix domain, we have to solve the following set of problems:

$$\begin{aligned} &\min_{x_{\searrow j} \in \mathcal{P}_{\searrow j}} Q_j[\phi_{j+1}(x_j, x_{\searrow j})] \quad \text{and} \\ &\max_{x_{\searrow j} \in \mathcal{P}_{\searrow j}} Q_j[\phi_{j+1}(x_j, x_{\searrow j})] \end{aligned} \tag{19}$$

Here, ϕ_{j+1} represents the postfix or remainder of the formula, i.e., a function that only captures the dependence of the current variable and the prefix and abstracts over the postfix variables $x_{j+1}, \ldots, x_n$. In the simplest case, this corresponds to the calculation of upper and lower bounds for a given leaf node within the decision tree of the previous section. In that situation, Q_j would correspond to a randomized quantifier and ϕ_{j+1} is the known function ϕ of Eq. (1). To estimate these upper and lower bounds, we use the combination of importance sampling and interval propagation as detailed in the previous section. To be able to apply this procedure in the nested setting, we assume that we can draw a sample interval function

$$\left(\phi^i_{j+1} : \mathcal{P}^j_{h,i} \times \mathcal{P}_{\searrow j} \to \left(\mathbb{R}^+\right)^2\right) \sim \pi_{\phi_{j+1}}$$

from a distribution $\pi_{\phi_{j+1}}$ whose support function ϕ^s_{j+1} is also known. Thus, the samples (indicated by the index i) of $\pi_{\phi_{j+1}}$ can be evaluated at any $(x_j, x_{\searrow j})$:

$$\begin{aligned} &\phi^i_{j+1}(x_j, x_{\searrow j}) = [\underline{\phi}^i_{j+1}(x_j, x_{\searrow j}), \overline{\phi}^i_{j+1}(x_j, x_{\searrow j})] \\ &\phi^s_{j+1}(x_j, x_{\searrow j}) \text{ s.t. :} \\ &\qquad P\left(\phi^i_{j+1}(x_j, x_{\searrow j}) \subset \phi^s_{j+1}(x_j, x_{\searrow j})\right) = 1 \ \forall i \end{aligned}$$

For the distribution $\pi_{\phi_{j+1}}$ to be constructed below, we will show that it has the desirable property of giving conservative confidence intervals for the probability of satisfaction outlined in Eq. (19). These quantities will be memorized as functions, hence the conservative confidence intervals can be calculated for arbitrary values of confidence levels.

With this representation, the general procedure of the modified algorithm consists of the path selection and domain splitting, drawing samples and finally updating the quantities at the nodes within the decision tree using the information obtained from these samples (see Algorithm 1).

4.2.2 Path selection and domain splitting

Given a forest of decision trees, the selection of a path along the forest is nearly unchanged compared to the HOO algorithm. That is, in analogy to the HOO algorithm and the one presented in [7], a child is selected according to its upper and lower confidence bounds (denoted by $\underline{B}^j_{h,i}, \overline{B}^j_{h,i}$) for existential and universal quantifiers, respectively. If there is no unique maximal (for existential quantifiers) or minimal (for universal quantifiers) branch, among the optimal branches one is selected at random (uniformly). In the case of a randomized quantifier, we choose a child at random, uniformly distributed across all children, thereby guaranteeing a uniform proposal distribution q for the importance sampling estimate in Eq. (10), see below. After this path selection process for each variable, we have selected a path within its cover tree each of which has a smallest sub-domain.

Within the HOO algorithm, there is only one cover tree representing the domain of the variable to optimize over, hence the selected path leading to a leaf of this cover tree has the smallest domain at the leaves. This sub-domain is split into two sub-domains thereby refining the region which should be explored next. As the selected path in the nested setting contains multiple domains, we select one variable, *i.e.,* one cover tree within the tree of trees, for which we refine the smallest domain analogously to the HOO algorithm. Among all variables within the SSMT formula, we decide for the variable which has the largest domain at its leaf node along the selected path through the decision tree. Note that for cover trees corresponding to a quantifier over a finite domain, the leafs (smallest sub-domains) contain only a single point. These quantifiers are never selected for splitting the domain.

In addition to splitting the sub-domain, we have to deal with the part of the decision tree below the node which is selected for refinement (for example, in Fig. 5 the sub-tree with the existential quantifier over x_2 as root node). To this end, we create a copy of this sub-tree such that each of the leaf nodes after a refinement is linked to one copy (see Fig. 6 for the resulting tree after a refinement of the root node, which in this example is also a leaf node of the first cover tree for variable x_1). As samples must not be copied—it would violate the independence assumption of the drawn samples, *i.e.,* we would use a single sample multiple times—we only copy the support calculation for the sub-formulas associated with the nodes within the sub-tree and delete the individual

samples. Note that although the prefix of the copy of the sub-tree changes, the support for values of the corresponding sub-formula is still valid, i.e., gives an over-approximation of the actual support, as it was valid for the larger domain and by splitting the prefix domain values can only shrink. Instead of only splitting the largest sub-domain, we could also divide every sub-domain within the selected path. However, for all results shown in Sect. 5, we only split the largest sub-domain. The final selected path after the splitting then results from concatenating the selected prefix, selecting one of the newly created sub-domains at random (uniformly distributed), and adding the selected path before splitting within the postfix.

4.2.3 Draw a sample

Within the selected path of nodes from the root of the decision tree to the leaves, we have sampled sub-domains for the randomized quantifier. To obtain a point sample which is uniformly distributed across the complete domain of the randomized quantifier, we additionally sample a point from the selected leaf of a randomized quantifier. For reference, let the kth sample point at the randomized quantifier Q_j be denoted with $\tilde{x}_j^k$. This point sample is in turn used to generate the random functions by valuating variable x_j within the function ϕ and density functions using this sample, but leaving the dependence of the other variables symbolically. As samples are drawn uniformly across the domain instead of using the original probability distribution, we have to correct for this by multiplying the random function by the importance weight given by the density of the uniform distribution $\frac{1}{|\mathcal{P}_{h,i}^j|}$ [see also Eq. (17)].

4.2.4 Updates

Starting at the bottom of the decision tree, we iteratively use the point samples drawn at the previous stage to construct an interval function sample for each quantifier along the selected path, which in turn can be used for upper quantifiers within the path and thereby constructing the distribution of interval samples. In the simplest case of a leaf node at the very bottom of the decision tree, the sample of the sub-tree would consist of the interval function ϕ from Eq. (1). In that case, the individual samples and the support function would be identical. Assuming that we have drawn such an interval sample for the postfix quantifier ϕ_{j+1}^i, we can construct upper and lower empirical averages μ, respectively, which in turn can be used to set upper and lower U values, similar to the HOO algorithm described in the previous section. That is, depending on the current quantifier Q_j, we have the following definition of lower and upper bounds for the empirical averages:

$$
\begin{aligned}
[\mu_{h,i}^{j-}, \mu_{h,i}^{j+}] &= \begin{cases} \frac{1}{T_{h,i}^j} \sum_{k=1}^{T_{h,i}^j} \phi_{j+1}^k(\mathcal{P}_{h,i}^j, \mathcal{P}_{\searrow j}) & \text{if } Q_j \in \{\forall_{x_j} \exists_{x_j}\} \\ \frac{1}{T_{h,i}^j} \sum_{k=1}^{T_{h,i}^j} \frac{\phi_{j+1}^k(\tilde{x}_j^k, \mathcal{P}_{\searrow j}) p(\tilde{x}_j^k | \mathcal{P}_{\searrow j})}{\frac{1}{|\mathcal{P}_{h,i}^j|}}, \ \tilde{x}_j^k \sim \mathcal{U}_{\mathcal{P}_{h,i}^j} & \\ \text{if } Q_j = \text{Ʉ}_{x_j} \end{cases} \\
\phi_{j+1}^k(\mathcal{P}_{h,i}^j, \mathcal{P}_{\searrow j}) &\supset \Bigg[\min_{x_j, x_{\searrow j} \in \mathcal{P}_{h,i}^j \times \mathcal{P}_{\searrow j}} \phi_{j+1}^k(x_j, x_{\searrow j})^-, \\
&\quad \max_{x_j, x_{\searrow j} \in \mathcal{P}_{h,i}^j \times \mathcal{P}_{\searrow j}} \phi_{j+1}^k(x_j, x_{\searrow j})^+ \Bigg] \\
\phi_j^k(\underbrace{x_{j-1}, x_{\searrow (j-1)}}_{x_{\searrow j}}) &:= \begin{cases} \phi_{j+1}^k\left(\mathcal{P}_{0,1}^j, x_{\searrow j}\right) \text{ if } Q_j \in \{\forall_{x_j} \exists_{x_j}\} \\ \phi_{j+1}^k\left(\tilde{x}_j^k, x_{\searrow j}\right) p(\tilde{x}_j^k | \mathcal{P}_{\searrow j}) |\mathcal{P}_{0,1}^j| \\ \text{if } Q_j = \text{Ʉ}_{x_j} \end{cases} \\
\phi_j^s(x_{\searrow j}) &\supset \left[\min_{x_j \mathcal{P}_{h,i}^j} \phi_{j+1}^s(x_j, x_{\searrow j})^-, \max_{x_j \in \mathcal{P}_{h,i}^j} \phi_{j+1}^s(x_j, x_{\searrow j})^+ \right]
\end{aligned}
\tag{20a}
$$

Here, as we assumed to have access to the symbolic form of the functions involved (in particular the conditional density function $p(x_j | x_{\searrow j})$), an upper and lower bound on the maximum and minimum, respectively, in the last equation can be calculated using interval arithmetic. The equation for the randomized quantifier also recursively defines the distribution from which we can draw sample interval functions for sub-trees/quantifiers above the current node using importance sampling. Specifically, one first draws a point $\tilde{x}_j^k$ from the (sub-)domain of the current quantifier according to a proposal distribution, which we assumed here to be the uniform distribution. Using this point sample, we can weight the sample of the postfix tree ϕ_{j+1}^k using the importance weight $p(\tilde{x}_j^k | \cdot) \cdot |\mathcal{P}_{h,i}^j|$ to obtain a sample function, which can be evaluated at any prefix domain. That is, the kth term in the sum of the above equation can be used as a sample for the jth quantifier. For the quantifiers other than the randomized one, we evaluate the sample from the postfix on the whole sub-domain $\mathcal{P}_{h,i}^j$ using interval propagation to obtain an interval sample with the desired properties outlined in the previous section. Given these empirical averages, we can construct upper and lower bounds U analogously to the HOO algorithm, which require to hold with a confidence of at least $1 - 2\delta$:

$$
\begin{aligned}
[\underline{U}_{h,i}^j(\delta), \overline{U}_{h,i}^j(\delta)] &= [\underline{\mu}_{h,i}^j, \overline{\mu}_{h,i}^j] \\
&\pm \begin{cases} |\phi_{j+1}^s(\mathcal{P}_{h,i}^j, \mathcal{P}_{\searrow j})| \sqrt{-\frac{\log(\delta)}{2T_{h,i}^j}} : Q_j \in \{\exists, \forall\} \\ |\phi_{j+1}^s(\mathcal{P}_{h,i}^j, \mathcal{P}_{\searrow j})| |p(\mathcal{P}_{h,i}^j | \mathcal{P}_{\searrow j})| |\mathcal{P}_{h,i}^j| \\ \sqrt{-\frac{\log(\delta)}{2T_{h,i}^j}} : Q_j = \text{Ʉ} \end{cases}
\end{aligned}
\tag{21}
$$

Here $|\cdot|$ denotes the width of the support and width of the domain, respectively. Calculating the support of the importance weights in the case of a randomized quantifier can again be achieved using interval arithmetics, as we have assumed to have symbolic access to the involved functions. These bounds give a uniform (over the whole prefix domain) bound on the expected values and therefore provide bounds on the maximum and minimum with the necessary level of confidence. Specifically, the following holds:

Lemma 1 *Let $U^j_{h,i}(\delta)$ be defined as in Eq. (21), then the following holds:*

$$P\left(\overline{U}^j_{h,i} > \max_{\mathcal{P}_{\searrow j}} \mathbb{E}[\phi_{j+1}(x_1,\ldots,x_n)]\right) \geq 1-\delta$$

Note if we chose $\delta = n^{-4}$, we obtained the choice of the HOO algorithm in Eq. (5c). However, if the choice of the confidence level depends on the overall number of evaluations n, the U values would have to be updated within the whole tree, as δ enters directly the defining Eq. (21). Therefore, we directly set δ to a fixed value from the start. As a consequence, we only need to update the values along the selected path. As the HOO algorithm subsequently samples individual points within the domain of the optimization variable, the resulting rewards are dependent and hence one has to show that the resulting sequence forms actually a Martingale. In contrast, we are sampling independently and uniformly from the domain of each randomized quantified variable [see Eq. (20a)]. Sampling uniformly from the domain can also be seen as a proposal distribution, rendering the estimation of the expected value into a (parametric) importance sampling-based estimate. In addition, the domain of each parameter in this expectation is considered on each individual sample. Thus, instead of applying Hoeffding's inequality [21], we could also use a potentially tighter bound, such as the empirical Bernstein inequality [22], thereby obtaining a uniform confidence bound on the true range of the underlying expected value as a function of the parameters involved.

To construct tighter bounds, *i.e., B* values, also in a similar fashion to the previous section, we have to combine the confidence bounds provided by the above equations for the different operations associated with the different quantifiers, thereby propagating confidence intervals from the bottom to the root of the decision tree (forest) [7]. To this end, we state the following Lemmas.

Lemma 2 *Assume $P\left(\mu_i \leq \hat{\mu}_i - \epsilon_i\right) \leq \delta, \quad i = 1,2$ and let $l_i := \hat{\mu}_i - \epsilon_i$. Then the following holds:*

$$P\left(\max\{\mu_1,\mu_2\} \leq \max\{l_1,l_2\}\right) \leq \delta$$

Analogously, we have for the upper bound

Lemma 3 *Assume $P\left(\mu_i \geq \hat{\mu}_i + \epsilon_i\right) \leq \delta, \quad i = 1,2$ and let $u_i := \hat{\mu}_i + \epsilon_i$. Then the following holds:*

$$P\left(\max\{\mu_1,\mu_2\} \geq \max\{u_1,u_2\}\right) \leq \delta$$

Therefore, we can use the maximum of the two U values of the two branches to construct a tighter B value bound on the maximum over the domain, represented by the two branches. Analogously, it can be shown that the same statements hold if the maximum is replaced by a minimum, thereby providing combination rules for a minimum operation, i.e., universal quantifier. Note that the actual form of the upper and lower bound estimates does not matter here. In fact any bound which holds up to the given confidences can be applied. That is, combining two U values or combining a U value with an already combined other bound result both in a valid bound on the maximum or minimum, respectively. Taken together, we update the B values, once we updated the U values, exactly as in the HOO case:

$$\begin{aligned} B^j_{h,i} &= \min\{U^j_{h,i}, \max\{B^j_{h+1,2i}, B^j_{h+1,2i+1}\}\} \text{ if } Q_j = \exists \\ B^j_{h,i} &= \max\{U^j_{h,i}, \min\{B^j_{h+1,2i}, B^j_{h+1,2i+1}\}\} \text{ if } Q_j = \forall \end{aligned} \tag{22}$$

For a randomized quantifier, we need to sum two confidence intervals, which we split in the sum of the upper bound and the lower bound, respectively. Note that in the standard HOO setting, the domain of the random variable is not split, hence there is no combination of bounds for this kind of operation. Unfortunately, the sum of two confidence intervals does not necessarily hold with the same confidence as the individual ones. Therefore, we distribute the confidence level to each of the individual ones to obtain the same overall confidence when multiplied. As the samples in each branch are assumed to be drawn independently, we have the following lemma for the sum of two confidence intervals.

Lemma 4 *Assume $P\left(\mu_i \geq \hat{\mu}_i + \epsilon_i\right) \leq \delta_i, \quad i = 1,2$ with $\delta_1 \cdot \delta_2 = \delta$ and let $u_i := \hat{\mu}_i + \epsilon_i$. Then the following holds:*

$$P\left(\mu_1 + \mu_2 \geq u_1 + u_2\right) \leq \delta$$

4.2.5 Stopping criterion

For a fixed level of confidence δ, we can iteratively select, split, sample, and update the decision tree to obtain confidence intervals, in particular at the root of the decision tree, which states intervals containing the true value up to the given confidence. These intervals only hold for a particular choice of iterations chosen in advance. However, oftentimes the decision maker would like to decide sequentially, as more information becomes available. To obtain a valid sequential

Algorithm 2 Calculating the confidence bounds for a node

```
function getConfidenceBounds(node, confidence)
    localConfidence ← HoeffdingBound(node, confidence)   ▷ See equation (21)
    if node ≙ ⊎ then
        n1 ← node.childLeft.n; n2 ← node.childRight. ▷ Number of evaluations
        1 − δ ← confidence
        (1 − δ1) ← (1 − δ)^(n1/(n1+n2)); (1 − δ2) ← (1 − δ)^(n2/(n1+n2))
        interval1 ← getConfidenceBounds(node.childLeft, (1 − δ1))
        interval2 ← getConfidenceBounds(node.childRight, (1 − δ2))
        propInterval ← interval1 + interval2
    else if node ≙ ∃ then
        interval1 ← getConfidenceBounds(node.childLeft, (1−δ))
        interval2 ← getConfidenceBounds(node.childRight, (1 − δ))
        propInterval ← max(interval1, interval2)
    else if node ≙ ∀ then
        interval1 ← getConfidenceBounds(node.childLeft, (1−δ))
        interval2 ← getConfidenceBounds(node.childRight, (1 − δ))
        propInterval ← min(interval1, interval2)
    end if
    return localConfidence ∩ propInterval
end function
```

hypothesis test, based on the proposed algorithm to calculate the confidence intervals for the root node, we chose the confidence levels to be $\delta = \frac{\Delta}{N}$ with Δ being the desired sequential confidence level and N denoting the maximal number of evaluations one is willing to perform in the worst case. This decision rule is known from the so-called racing algorithms (see [23]). Using this increased confidence level accounts for the fact that in a sequential setting the decision maker can make a wrong decision at any stage before the maximal number of evaluations with a probability bounded by some confidence level. As the number of these sequential decisions increases, the probability of making such a wrong decision also increases which is bounded by the sum of the probabilities of wrong decisions across the number of possible test, i.e., maximal number of evaluations.

Taken together, we have the following Algorithm 2 to combine the different estimates within the decision tree to a confidence interval which holds with a given level of confidence. Here, we have chosen a specific separation of confidences, depending on the number of evaluation of the two branches n_1, n_2. This particular choice offers a trade-off between the cases $n_1 = n_2$ and $n_1 = 0, n_2 > 0$ ($n_2 = 0, n_1 > 0$), for which optimal choices can be calculated.

4.3 Confidence intervals for nested algorithm

The main result of the algorithm lies in the confidence intervals at the root node, which can be computed at any given time during the execution of the algorithm. As mentioned, care has to be taken when this confidence interval is used as a sequential hypothesis test. Using the Lemmas from the previous section, we can state the main result in the following theorem.

Theorem 1 *Let $\Phi := Q_1 \dots Q_k : \phi(x_1, \dots, x_k)$ be a given SSMT formula. Further, let $B_n(\delta) = [\underline{B}_n(\delta), \overline{B}_n(\delta)]$ be the estimated interval of the root node of the decision tree corresponding to Φ after n evaluations obtained from Algorithm 2 and let $0 \leq \delta \leq 1$ be a given number. Then, B_n contains the true probability of satisfaction (quantitative value of the SSMT formula) with probability (over the sample space) of at least $1 - \delta$:*

$$P\left(\Phi \in B_n(\delta)\right) \geq 1 - \delta$$

Proof The local confidence intervals are calculated using Eq. (21). If there are children descending from the current node, this local confidence interval is returned by the algorithm. As the corresponding samples are obtained using independent samples of the uniform proposal distribution, and the support of the samples are safely over-approximated using interval arithmetics, Hoeffindg's inequality can be applied, so that the local confidence intervals are indeed confidence intervals subject to the given confidence level δ. Again, due to the over-approximating interval of the support [see Eq. (20a)], Hoeffing's inequality holds uniformly over all specific values within the domains of the prefix to the current node. Using Lemmas 2–4, we know that the propagated intervals (propInterval within Algorithm 2) are also valid confidence intervals to the same level δ. Therefore, returning the conjunction of these two intervals gives a valid confidence interval. □

5 Evaluation

5.1 Path planning use case

To illustrate the potential of the approach on a use case, we consider the following stochastic hybrid verification task for a simplified decision making unit within the domain of autonomous driving. In particular, we consider the following scenario. In a constrained driving situation on an expressway, an ego-vehicle (green car in Fig. 7) is forced to decide for one of two potential options:

1. Staying on the right lane
2. Changing lanes

Both options have an associated risk of a collision, as the vehicle in front of the ego-vehicle is considered to have very low speed (∼0) whereas the car behind (blue vehicle)

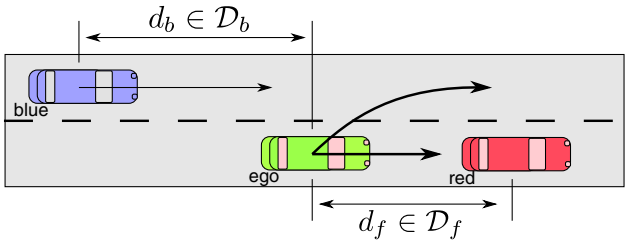


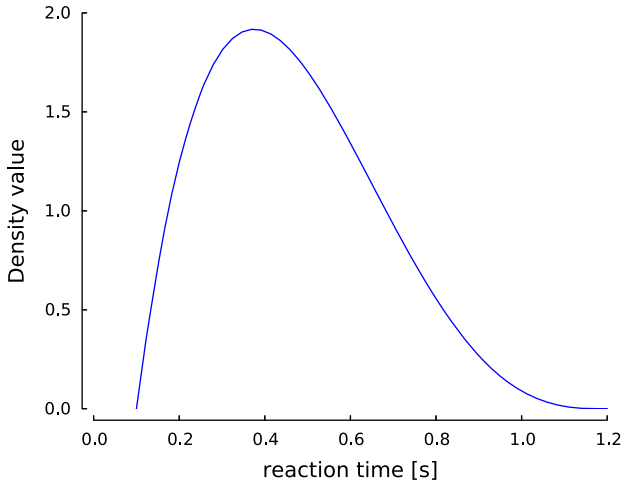
Fig. 7 Illustration of the path planning use case. The green Ego-vehicle has to decide to either stay on the *lane* or *switching lanes* depending on its measurements of the distance to the other vehicles which might be subject to measurement noise

has very high speed. Therefore, depending on the two distances to the vehicles, both options could lead to a collision. If the ego-vehicle decides for a lane-change maneuver, the blue vehicle behind is assumed to decelerate in order to mitigate a potential collision. The situation is further complicated by the fact that the driver of the blue car is assumed to have a stochastic reaction time after which an also stochastic deceleration value is applied. Furthermore, we assume a non-deterministic inaccuracy within the sensors of the ego-vehicle such that the relative positions of the two vehicles are not exactly perceived by the ego-vehicle. Within this scenario, we are then interested in the existence of a best choice between the two potential decisions. Here, a choice is considered to be better based on an additional cost function which checks for collisions within a given setting. If a setting (pair of distances, fix reaction time, fix deceleration, fixed decision choice and fixed deviations of positions due to sensor inaccuracies) leads to a collision, then the cost function assigns higher costs to settings which have a larger difference in velocities during collision. That is a collision between cars running at $50\frac{\text{km}}{\text{h}}$ and $100\frac{\text{km}}{\text{h}}$, respectively, is more costly than a collision between cars at $50\frac{\text{km}}{\text{h}}$ and $51\frac{\text{km}}{\text{h}}$. The existence of a best choice can be formalized using the following SSMT formula:

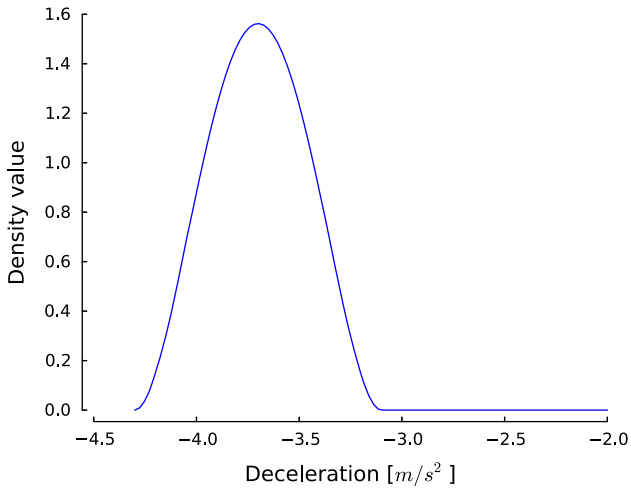
$$\forall_{d_b \in \mathcal{D}_b} \forall_{d_f \in \mathcal{D}_f} \mathbin{\mathsf{U}}_{\Delta t \sim \beta(\alpha_t, \beta_t)} \mathbin{\mathsf{U}}_{a_b \sim \beta(\alpha_a, \beta_a)} \exists_{i \in \{0,1\}} \forall_{\epsilon_p \in \Delta_p} \forall_{\epsilon_v \in \Delta_v} : c_i(d_b, d_f, \Delta_t, a_b, \epsilon_p, \epsilon_v) < c_{\{i\}^c}(d_b, d_f, \Delta_t, a_b, \epsilon_p, \epsilon_v) \tag{23}$$

Within this formula, we assumed a beta distribution for both the distribution of reaction times of the blue vehicle's driver and the distribution of deceleration values she chooses after the reaction time. For the sake of simplicity, the beta distributions are characterized by fixed shape and scale parameters α and β. Note that the approach would also allow for these parameters to depend on the two distances d_f and d_b. The particular choice of distributions is shown in Fig. 8.

ϵ_p and ϵ_v are the deviations in positional and velocity sensor measurements due to internal inaccuracies. To compute



(a) Reaction time, $\alpha = 2$, $\beta = 4$



(b) Deceleration, $\alpha = 3$, $\beta = 3$

Fig. 8 Reaction time and random deceleration distribution.

the costs incurred with either of the two decision options, we use a simple quadratic dynamic model, assuming piecewise constant acceleration:

$$s_t = s_0 + t v_0 + \frac{1}{2} t^2 a_0 \tag{24}$$

Here, s_t represents the different positions of each of the vehicles, and v_0 and a_0 are their velocities and accelerations, respectively, at the time a decision is drawn ($t = 0$). If the ego-vehicle is opting for the lane change, the acceleration for the blue vehicle is set to a_b after its reaction time Δ_t. If for a given setting characterized by the parameter combination $(d_b, d_f, \Delta_t, a_b, \epsilon_p, \epsilon_v)$, a collision is detected between vehicles having velocities v_1 and v_2, and the incurred collision costs are defined to be:

$$c(d_b, d_f, \Delta_t, a_b, \epsilon_p, \epsilon_v) = |v_1 - v_2| \max(v_1, v_2) \tag{25}$$

This choice was made to account for the fact that impacts at higher velocities are more severe than at lower speeds. However, note that any other choice of cost function would also be suitable. Intuitively the formula in Eq. (23) expresses the minimal probability across distances that there exists an option which is guaranteed to be the safer choice under all possible sensor measurements, i.e. environments which are consistent with the sensor data.

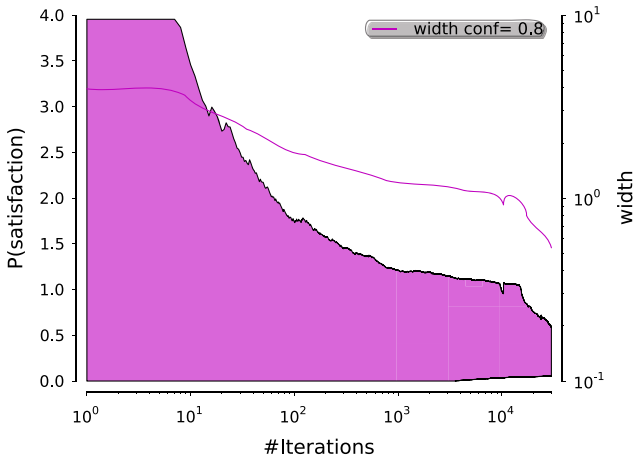
To evaluate this formula, we used the interval-based algorithm presented in the previous sections. Note that the last three quantifiers are non-random and can therefore be handled using a standard SAT modulo theory solver addressing the relevant fragment of arithmetic. In fact, analogously to [7], we evaluated this part of the formula completely, again using interval arithmetics. To this end, we have to determine the interval of potential collision speeds, which in turn amounts to calculating first and last possible points in time at which a collision could happen opting for either option (lane change or staying on the lane). If the two costs for the different options do not overlap, we know that there is a guaranteed safer option for the particular choice of distances, reaction time, and deceleration. If the cost intervals overlap, and the intervals specifying distances and reaction time are below a critical value, we accept both options as sufficiently safe. In Fig. 9, we plotted the evolution of the confidence interval at the root, i.e., the confidence interval for the overall probability of satisfaction for three difference confidence levels. Note that although we know that the overall probability of satisfaction has to be between zero and one, the estimated confidence interval is for relatively few samples larger than this interval. We could have incorporated this additional knowledge; however, for illustration purposes, we decided not to cut this interval. This is partly because of the additional slack one has to add due to Hoeffding's inequality as can be seen by the larger confidence interval for higher confidence levels. Furthermore, the interval arithmetics only give an over-approximation of the true values, hence the additional slack at the beginning, when the branching has not progressed and therefore the individual sub-regions are quite large. As can be seen from the right panel in Fig. 9, when determining the minimal probability, only one variable d_b seems to have a significant effect, as the frequency of evaluating sub-regions of d_f does not vary across its domain, indicating a flat mean function over the considered range.

5.2 Comparison with deterministic abstraction approach

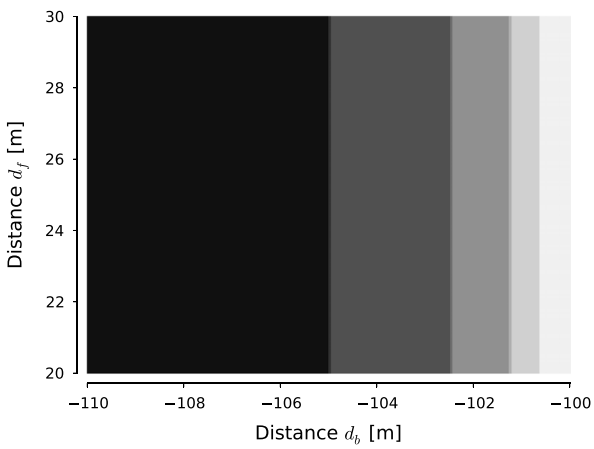
Most other approaches to solve hybrid-state Markov decision processes are based on abstraction approaches to evaluate continuous problems as the ones studied in the previous sections (see [12,24,25]). To compare the presented algorithm with such an approach, we use the simple example introduced in Sect. 3.1. The corresponding SSMT formula encoding this example is:

$$\exists_{x\ in[-5,5]} \mathbb{H}_{y|x\sim\mathcal{N}(x,20+x^2)}(x^2+y^2 \leq 10) \tag{26}$$

To obtain a valid abstraction which can be used in other model checkers, we equidistantly discretize both the domain of the existentially as well as of the randomized quantified variable. For each cell within the grid, we can calculate an

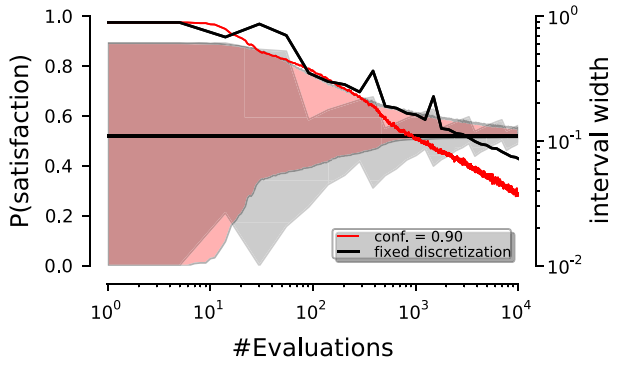


(a) Confidence Interval Evolution

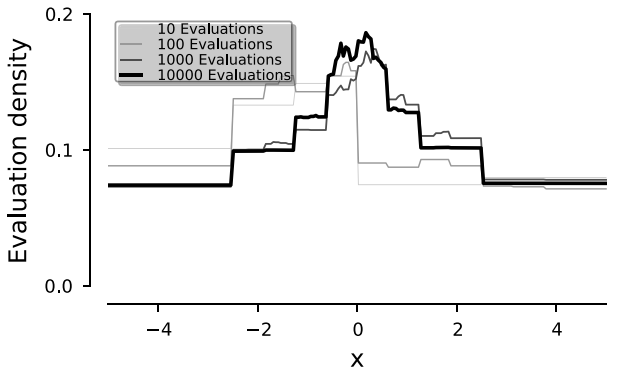


(b) Histogram over selected sub-regions. Only d_b seems to have a significant effect, as the frequency of evaluating sub-regions of d_f does not vary across its domain, indicating a flat mean function over the considered range.

Fig. 9 *Left panel* Evaluation of the autonomous car decision problem. The figure displays the confidence interval for the probability of satisfaction as a function of number of iterations. *Right panel* Distribution of distances explored for the autonomous car decision problem during the evaluation. The figure displays the frequency after the 3,000 iterations. Brighter regions indicate more frequent selection

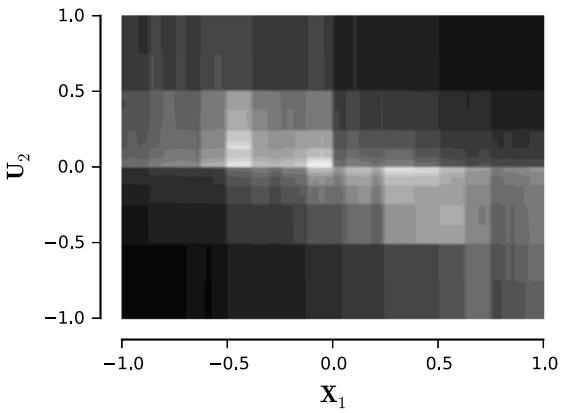


(a) Comparison of the interval-width

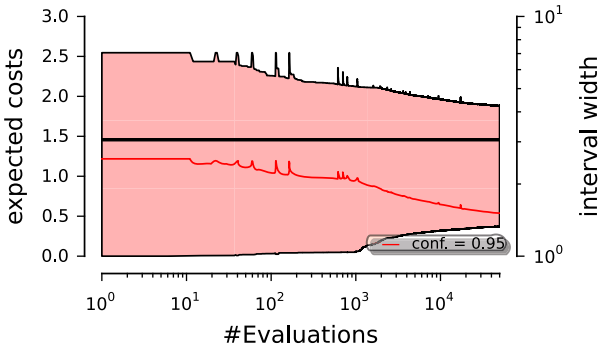


(b) Histogram of the selected location for the statistical algorithm

Fig. 10 Comparison of the presented algorithm and an abstraction approach using a fixed discretization abstraction approach. Shown is the width of the obtained (confidence) interval for the probability of satisfaction as a function of evaluations or cumulative number of grid cells of the abstraction, respectively. Results for the statistical evaluation are averaged across 60 iterations. In the lower panel, we show a histogram representing the frequency of selecting a sub-region for further exploration



(a) Frequency of evaluations for different domain combinations estimated across 70000 iterations, where brighter regions indicate a more frequent selection.



(b) Convergence of the estimated interval for different confidences.

Fig. 11 Evolution of the confidence intervals and *histogram* of the explored state–action combinations

upper and lower bound on the probability falling into a given cell in the same way as we did in Eq. (7). A deterministic Model Checker then evaluates the complete discretization tree and thereby obtains guaranteed upper and lower bounds. The accuracy of this bound, however, depends on the quality of the abstraction, that is, the resolution of the discretization in this example. To compare the efficiency of the two approaches in a sequential setting, we compare the width of the obtained bound on the probability of satisfaction as a function of the number of evaluations and the number of cumulative number of grid cells in Fig. 10a. We chose the cumulative number of grid cells to represent the effort of a fixed discretization in a sequential setting, in which a decision maker has the option to refine the discretization to get a more accurate bound on the probability of satisfaction at the cost of re-evaluating the whole domain at a finer resolution. In addition, we plotted in the lower panel the frequency with which a sub-region is selected for further exploration at different stages of the evaluation process. As can be seen, although the confidence for the individual nodes during the execution of the algorithm is very close to 1 (as the maximal number of evaluations was set to 10^4), the obtained confidence intervals are consistently tighter than the corresponding guaranteed intervals resulting from an equidistant discretization with a corresponding abstraction approach.

5.3 Stochastic optimal control

To illustrate the behavior in a more complex setting, we adopt a simple stochastic optimal control example, for which the optimal solution can still be calculated analytically:

$$\mathbf{X}_{t+1} = \boldsymbol{A}\mathbf{X}_t + \boldsymbol{B}\mathbf{U}_\mathbf{t} + \Sigma\boldsymbol{\epsilon}, \qquad \boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbb{1}) \tag{27a}$$

$$C(\mathbf{X}_t, \mathbf{U}_t) = \mathbf{X}_t^\top \boldsymbol{R}_x \mathbf{X}_t + \mathbf{U}_t^\top \boldsymbol{R}_u \mathbf{U}_t \tag{27b}$$

Here, $\mathbf{X}$ represents a state vector, which evolves over time according to the linear stochastic dynamics (Eq. (27a)) under the influence of a control input $\mathbf{U}$. The task of optimal control

now consists of minimizing the expected costs C over control actions. Specifically, in a closed loop, bounded horizon setting (here, we consider only 2 consecutive time steps), the task can be formalized as a nested optimization–expectation problem of the following form:

$$\min_{\mathbf{U}_0} \mathbb{E}_{\mathbf{X}_1|\mathbf{U}_0,\mathbf{X}_0}\left[\min_{\mathbf{U}_1}\left\{\mathbb{E}_{\mathbf{X}_2|\mathbf{U}_1,\mathbf{X}_1}\left[\sum_t C(\mathbf{X}_t,\mathbf{U}_t)\right]\right\}\right] \quad (28)$$

This corresponds to the following SSMT formula:

$$\forall_{\mathbf{U}_0} \exists_{\mathbf{X}_1|\mathbf{X}_0,\mathbf{U}_0} \forall_{\mathbf{U}_1} \exists_{\mathbf{X}_2|\mathbf{X}_1,\mathbf{U}_1} : \underbrace{C(\mathbf{X}_1,\mathbf{U}_0)+C(\mathbf{X}_2,\mathbf{U}_1)}_{\phi(\mathbf{X}_1,\mathbf{U}_0,\mathbf{X}_2,\mathbf{U}_1)} \quad (29)$$

To simplify things even further, we choose all involved matrices to be set to the identity matrix. The optimal control action in this case can be calculated to be a function of the current state $\mathbf{X}_t$ and is given by $\mathbf{U}_t^* = -\mathbf{X}_t$. With this action applied, we arrive at the optimal costs given by sum of the variances over the horizon to be considered:

$$C^* = \mathbb{E}[X_1^2] + \mathbb{E}[X_1^2] + \mathbb{E}[X_2^2] = 3 \quad (30)$$

To apply the algorithm presented in the previous sections, we have to constrain the different variables to a bounded domain. By doing so, the optimal control cannot be calculated analytically anymore. However, if the domain is sufficiently large and in particular encloses the origin, we can obtain a lower bound to the optimal control cost, which should be close to the optimal costs.

As can be seen from Fig. 11a, the algorithm tends to evaluate combinations which are close to the optimal control strategy ($U_2 = -X_1$) more often. In the lower panel 11b, we have shown the evaluation of the estimated expected costs, i.e., the evaluation of the SSMT formula in Eq. (29), as a function of iterations. Although the estimated interval decreases around the true value (black horizontal line), the convergence is rather slow reflecting the complexity of the evaluation problem.

6 Conclusion

We presented an algorithm based on statistical model checking which can handle combinations of demonic nondeterminism and stochastic behavior in quite general settings, including, but not limited to, the classical example of Markov decision processes. Within such a Markovian setting other approaches such as solving the Hamilton–Jacobi–Bellman equation are particularly designed for handling continuous stochastic optimal control problems. However, these approaches typically assume a simple form of the cost functions and do not scale well, as the discretization of the partial differential equation has to be chosen beforehand. We presented an approach that is applicable to more general problems, not necessarily being Markovian. One of its features is an adaptive discretization of the state space, where more promising regions are explored more frequently and thereby split into sub-regions. The approach is based on a known algorithm from noisy optimization, namely HOO. Our approach however differs from HOO in some aspects. First, instead of applying a potentially pessimistic bound based on the Lipschitz constant of the reward function, we are using interval propagation method to obtain uniform confidence bounds on the expected values for the nodes within the cover tree. In addition, again using interval propagation methods, we can not only estimate confidence bounds but also compute the support of the individual samples used for the empirical averages with certainty. As a consequence, the width of the confidence bounds not only shrinks with additional samples but also by refining the domains and hence narrowing the support of the samples. Nevertheless, as the confidence statements of the presented approach are mainly based on Hoeffding's inequality, it also suffers from a prohibitively large number of samples needed to show that probabilities of satisfaction are below very small thresholds like $\sim 10^{-9}$—unless the analytical determined support already gives the desired bound. A typical approach for these rare events is to use particularly tuned sampling techniques, such as importance sampling or importance splitting. As importance sampling is already covered within our approach, future research directions will explore the possibility to tune the proposal distribution, such that more accurate estimates for rare events can be obtained.

Acknowledgments The research leading to these results has received funding from the ARTEMIS Joint Undertaking under Grant Agreement No. 332830 (CRYSTAL) and German national funding from BMBF No. 01IS13001A, from the EU within the FP7 STREP "Modelling, verification and control of complex systems: From foundations to power network applications (MoVeS)", and by Deutsche Forschungsgemeinschaft DFG through the Transregional Coordinate Research Action SFB/TR 14 AVACS. In addition, we would like to thank DENSO Automotive Deutschland GmbH and DENSO Corp. for kindly providing the path planning use case. In particular, we would like to thank M. Toyoshima, M. Adachi and B. Böddeker for many fruitful discussions.

Appendix: Proof of lemmas

Proof of Lemma 1

Proof We only consider the upper bound, the argument for the lower bound is analogous. We first consider the case of an existential or universal quantifier. Within the construction of the empirical means, we used interval arithmetic to obtain the maximum and minimum over the whole domain $\mathcal{P}_{h,i}^j, \mathcal{P}_{\searrow j}$ for each individual sample [see Eq. (20a)]. That is the empirical mean can be written as:

$$\overline{\mu}^j_{h,i} = \frac{1}{T_{h,i}} \sum \max_{\mathcal{P}^j_{h,i}, \mathcal{P}_{\searrow j}} \phi(x_j, x_{\searrow j})$$

Therefore, setting $\epsilon := |\phi^s_{j+1}(\mathcal{P}^j_{h,i}, \mathcal{P}_{-j})| \sqrt{-\frac{\log(\delta)}{2T^j_{h,i}}}$ and by applying Jensen's inequality (as the maximum is a convex operation), we have:

$$\begin{aligned} & P\left(\overline{\mu}^j_{h,i} + \epsilon > \max_{\mathcal{P}^j_{h,i}, \mathcal{P}_{\searrow j}} \mathbb{E}[\phi_{j+1}(x_1, \ldots, x_n)]\right) \\ & \geq P\left(\overline{\mu}^j_{h,i} + \epsilon > \mathbb{E}\left[\max_{\mathcal{P}^j_{h,i}, \mathcal{P}_{\searrow j}} \phi_{j+1}(x_1, \ldots, x_n)\right]\right) \\ & \geq 1 - \delta \end{aligned}$$

where the last inequality is a result of applying Hoeffding's inequality to the modified random variable $Z := \max_{\mathcal{P}^j_{h,i}, \mathcal{P}_{\searrow j}} \phi(x_1, \ldots, x_n)$. □

Proof of Lemma 2

Proof The above inequality can be shown for each of the following cases
Case $\arg\max\{\mu_i\} = \arg\max\{l_i\}$
Holds trivially by assumption.
Case $\arg\max\{\mu_i\} \neq \arg\max\{l_i\}$
Let $i^* = \arg\max\{\mu_i\}, k^* = \arg\max\{l_i\}$. As $\mu_{k^*} \leq \mu_{i^*}$, the set of estimators l_{k^*} for which $\mu_{i^*} \leq l_{k^*}$ is included in $\mu_{k^*} \leq l_{k^*}$. Therefore, we have

$$\begin{aligned} P(\max\{\mu_1, \mu_2\} \leq \max\{l_1, l_2\}) &= P(\mu_{i^*} \leq l_{k^*}) \\ &\leq P(\mu_{k^*} \leq l_{k^*}) \leq \delta \end{aligned}$$

□

Proof of Lemma 3

Proof As above, we consider the following proof-by-cases:
Case $\arg\max\{\mu_i\} = \arg\max\{u_i\}$
Holds trivially by assumption.
Case $\arg\max\{\mu_i\} \neq \arg\max\{u_i\}$
Let $i^* = \arg\max\{\mu_i\}, k^* = \arg\max\{u_i\}$. Therefore, we have

$$\begin{aligned} & 1 - P(\max\{\mu_1, \mu_2\} \leq \max\{u_1, u_2\}) \\ & = 1 - P(\mu_{i^*} \leq u_{k^*}) \\ & \geq 1 - P(\mu_{k^*} \leq u_{k^*}) \geq 1 - \delta \\ & \Rightarrow P(\max\{\mu_1, \mu_2\} \geq \max\{u_1, u_2\}) \leq \delta \end{aligned}$$

□

References

1. Groote, J.F., van Vlijmen, Sebastiaan F.M., Koorn, Jan W.C.: The safety guaranteeing system at station hoorn-kersenboogerd. In: Proceedings of the Tenth Annual Conference on Computer Assurance (COMPASS), IEEE, pp 57–68 (1995)
2. Audemard, G., Bozzano, M., Cimatti, A., Sebastiani, R.: Verifying industrial hybrid systems with mathsat. Electron Notes Theor Comput Sci **119**(2), 17–32 (2005)
3. Sproston, J.: Model checking for probabilistic timed and hybrid systems. Ph.D. thesis, School of Computer Science, The University of Birmingham (2001)
4. 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.) Hybrid Systems: Computation and Control. Lecture Notes in Computer Science, vol. 4981, pp. 172–186. Springer, Berlin, Heidelberg (2008)
5. Littman, M.L., Majercik, S.M., Pitassi, T.: Stochastic boolean satisfiability. J. Autom. Reason. **27**(3), 251–296 (2001)
6. Teige, T., Eggers, A., Fränzle, M.: Constraint-based analysis of concurrent probabilistic hybrid systems: an application to networked automation systems. Nonlinear Anal. Hybrid Syst. **5**(2), 343–366 (2011)
7. Ellen, C., Gerwinn, S., Fränzle, M.: Confidence bounds for statistical model checking of probabilistic hybrid systems. In: Proceedings of Formal Modeling and Analysis of Timed Systems, Springer, Heidelberg, pp. 123–138 (2012)
8. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: Proceedings of Machine Learning: ECML, Springer, Berlin, Heidelberg, pp. 282–293 (2006)
9. Blom, H.A.P., Lygeros, J., (eds.): Stochastic Hybrid Systems: Theory and Safety Critical Applications, vol. 337. Springer, Heidelberg (2006)
10. Bubeck, S., Munos, R., Stoltz, G., Szepesvari, C.: X-armed bandits. J. Mach. Learn. Res. **12**, 1655–1695 (2011)
11. Fränzle, M., Herde, C.: HySAT: an efficient proof engine for bounded model checking of hybrid systems. Form. Methods Syst. Des. **30**(3), 179–198 (2007)
12. Fränzle, M., Hahn, E.M., Hermanns, H., Wolovick, N., Zhang, L.: Measurability and safety verification for stochastic hybrid systems. In: Caccamo, M., Frazzoli, E., Grosu, R. (eds.) HSCC, ACM, pp 43–52 (2011)
13. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. Inf. Comput. **94**(1), 1–28 (1991)
14. Sen, K., Viswanathan, M., Agha, G.: Statistical model checking of black-box probabilistic systems. In: Alur, R., Peled, D. (eds) Computer Aided Verification, Lecture Notes in Computer Science, vol. 3114. Springer, Berlin, Heidelberg, pp. 399–401 (2004)
15. Younes, H.L.S.: Ymer: a statistical model checker. In: Etessami, K., Rajamani, S. (eds.) Computer Aided Verification, Lecture Notes in Computer Science. vol. 3576. Springer, Berlin, Heidelberg, pp 171–179 (2005)
16. David, A., Larsen, K., Legay, A., Mikučionis, M., Poulsen, D., van Vliet, J., Wang, Z.: Statistical model checking for networks of priced timed automata. In: Fahrenberg, U., Tripakis, S. (eds.) Formal Modeling and Analysis of Timed Systems. Lecture Notes in Computer Science, vol. 6919. Springer, Berlin, Heidelberg, pp. 80–96, (2011)
17. Zuliani, P., Platzer, A., Clarke, E.M.: Bayesian statistical model checking with application to stateflow/simulink verification. In: Johansson, K.H., Wang Y. (eds.) Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control, ACM, Stockholm, Sweden, pp. 243–252 (2010)
18. Henriques, D., Martins, J.G., Zuliani, P., Platzer, A., Clarke, E.M.: Statistical model checking for markov decision processes. In: Pro-

ceedings of Quantitative Evaluation of Systems (QEST), 2012 Ninth International Conference on IEEE, pp. 84–93, (2012)
19. Fränzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. J. Satisf. Boolean Model. Comput. **1**(3–4), 209–236 (2007)
20. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2), 235–256 (2002)
21. Hoeffding, W.: Probability inequalities for sums of bounded random variables. J. Am. Stat. Assoc. **58**(301), 13–30 (1963)
22. Audibert, J.-Y., Bubeck, S., Munos R.: Bandit view on noisy optimization. In: Prfoceedings of Optimization for Machine Learning, MIT Press, pp 1–23 (2011)
23. Maron, O., Moore, A.W.: Hoeffding races: accelerating model selection search for classification and function approximation. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems 6. Morgan-Kaufmann, Burlington, MA, pp. 59–66 (1994)
24. Abate, A., D'Innocenzo, A., Di Benedetto, M.D.: Approximate abstractions of stochastic hybrid systems. Autom. Control IEEE Trans. **56**(11), 2688–2694 (2011)
25. Hahn, E.M..: Model checking stochastic hybrid systems. dissertation, Universität des Saarlandes (2013)