The Journal of Logic and Algebraic Programming 79 (2010) 436–466

Contents lists available at ScienceDirect

The Journal of Logic and Algebraic Programming

journal homepage: www.elsevier.com/locate/jlap

# Engineering constraint solvers for automatic analysis of probabilistic hybrid automata ☆,☆☆

Martin Fränzle *, Tino Teige, Andreas Eggers

*Dept. of Computing Science, Carl von Ossietzky Universität Oldenburg, D-26111 Oldenburg, Germany*

ARTICLE INFO

*Article history:*
Available online 15 July 2010

*Keywords:*
Probabilistic hybrid automata
Bounded model checking
Arithmetic constraint solving
Stochastic satisfiability

ABSTRACT

In this article, we recall different approaches to the constraint-based, symbolic analysis of hybrid discrete-continuous systems and combine them to a technology able to address hybrid systems exhibiting both non-deterministic and probabilistic behavior akin to infinite-state Markov decision processes. To enable mechanized analysis of such systems, we extend the reasoning power of arithmetic satisfiability-modulo-theories (SMT) solving by, first, reasoning over ordinary differential equations (ODEs) and, second, a comprehensive treatment of randomized (also known as stochastic) quantification over discrete variables as well as existential quantification over both discrete and continuous variables within the mixed Boolean-arithmetic constraint system. This provides the technological basis for a constraint-based analysis of dense-time probabilistic hybrid automata, extending previous results addressing discrete-time automata [33]. Generalizing SMT-based bounded model-checking of hybrid automata [5,31], stochastic SMT including ODEs permits the direct analysis of probabilistic bounded reachability problems of dense-time probabilistic hybrid automata without resorting to approximation by intermediate finite-state abstractions.



## 1. Introduction

Most embedded systems operate within or even comprise coupled networks of both discrete and continuous components. Safety assessment of such *hybrid systems* amounts to showing that the joint dynamics of the embedded system and its environment is well-behaved, e.g. that it may never reach an undesirable state or that it will converge to a desirable state, regardless of the actual disturbance. Disturbances may originate from uncontrolled inputs in an open system, like a car driver performing her driving task, as well as from internal sources of the overall technical system, like failing system components, including sensors or even actuators. Gradually advancing the capabilities of addressing such systems, research in hybrid system verification has thus traditionally focused on different classes of system structures and disturbances, ranging from a closed-system view over non-deterministic to probabilistic or stochastic hybrid systems. While the closed-system view necessitates a reasonably exact representation of the rather intricate yet deterministic feedback dynamics of coupled discrete and continuous systems, non-deterministic systems extend this view by unknown inputs of an open system. Probabilistic systems, finally, allow to capture unpredictable, yet statistically characterizable disturbances.

Within this article, we recollect different approaches to the constraint-based, symbolic analysis of hybrid systems and combine them to a technology able to deal with hybrid systems exhibiting both non-deterministic and probabilistic be-

☆ This article expands on the invited talk of the first author at the 20th Nordic Workshop on Programming Theory, NWPT 2008, Tallinn, 19–21 November 2008.

☆☆ This work was supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, www.avacs.org).

* Corresponding author.
*E-mail addresses:* fraenzle@informatik.uni-oldenburg.de (M. Fränzle), tino.teige@informatik.uni-oldenburg.de (T. Teige), andreas.eggers@informatik.uni-oldenburg.de (A. Eggers).

1567-8326/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.jlap.2010.07.003

havior as in infinite-state Markov decision processes, thus being able to model feedback behavior of hybrid systems with both unknown inputs or parameters and probabilistic sources of disturbance, like component failures. The resulting procedure enables constraint-based analysis of dense-time probabilistic hybrid automata, extending previous results addressing discrete-time probabilistic hybrid automata [33,61] and dense-time non-deterministic hybrid automata [29].

Our procedure belongs to the class of depth-bounded state-space exploration methods based on satisfiability solvers, which have originally been suggested for large finite-state systems by Groote et al. [36] and Biere et al. [16] and have since become popular under the term *bounded model checking* (BMC), now accounting for a major fraction of the industrial applications of formal verification. The idea of BMC is to encode the next-state relation of a system as a propositional formula, to unroll this to some given finite depth $k$, and to augment it with a corresponding finite unravelling of the tableaux of (the negation of) a temporal formula in order to obtain a propositional SAT problem which is satisfiable if and only if an error trace of length $k$ exists. Enabled by the impressive gains in performance of propositional SAT checkers in recent years, BMC can now be applied to very large finite-state designs.

Though originally formulated for discrete transition systems, the concept of BMC also applies to hybrid discrete-continuous systems. The BMC formulae arising from such systems comprise complex Boolean combinations of arithmetic constraints over real-valued variables, thus entailing the need for *satisfiability modulo theories* (SMT) solvers over arithmetic theories to solve them. Such SMT procedures are thus currently in the focus of the SAT-solving community (e.g. [8,19,28,32]), as is their application to and tailoring for BMC of hybrid system (e.g. [1,2,5,29,31]).

The scope of the aforementioned procedures, however, is confined to qualitative (i.e. non-probabilistic) models of hybrid behavior as well as to purely Boolean queries of the form "can the system ever exhibit an undesirable behavior?", whereas requirements for safety-critical systems frequently take the form of bounds on error probability, requiring the residual probability of engaging into undesirable behavior to be below an acceptable threshold. Automatically answering such queries requires, first, models of hybrid behavior that are able to represent probabilistic effects like component breakdown and, second, algorithms for state space traversal of such hybrid models.

In the context of hybrid systems augmented with probabilities, a wealth of models has been suggested by various authors. These models vary with respect to the patterns of continuous dynamics supported, the shapes of randomness modelled, and the degree to which they support non-determinism and compositionality. The cornerstones are formed by *probabilistic hybrid automata*, where state changes forced by continuous dynamics may involve discrete random events determining both the discrete and the continuous successor state [11,33,59], *piecewise deterministic Markov processes* [25], where state changes may happen spontaneously in a manner similar to continuous-time Markov processes, and *stochastic differential equations* [4], where, like in Brownian motion, the random perturbation affects the dynamics continuously. In full generality, stochastic hybrid system (SHS) models can cover all such ingredients [21,41]. While such models have a vast potential of application [17, 22,58], results related to their analysis and verification are limited, and often based on Monte-Carlo simulation [18,38]. For certain subclasses of piecewise deterministic Markov processes, of probabilistic hybrid automata, and of stochastic hybrid systems, reachability probabilities can be approximated [3,11,20,42] or, if continuous behavior is extremely restricted as in timed automata or the equivalent class of initialized rectangular hybrid automata, even computed [59,60].

For the automated state-exploratory analysis methods among the above, scalability is in general poor, as they are mostly relying on explicit representations of either bisimulation quotients or finite abstractions. In previous papers [33,61], we have presented a technology that preserved the virtues of SMT-based BMC, namely the constraint-based treatment of hybrid state spaces, while advancing the reasoning power to probabilistic models and requirements. While the technique is more general, these previous papers focus on depth-bounded reachability of *discrete-time* probabilistic hybrid automata, where the dynamics is given by (potentially non-linear) pre–post-relations rather than ordinary differential equations, as common in hybrid systems. Furthermore, assignments to continuous variables are confined to be deterministic in [33,61]. In the current submission, we strive to overcome these limitations by devising a procedure tackling dense time, dynamics described by (potentially non-linear) ordinary differential equations, and non-deterministic continuous behavior.

In order to achieve this goal, we first (in Section 2) define *dense-time probabilistic hybrid automata* (PHA) and provide their semantics as well as the bounded probabilistic reachability problem. We proceed in Section 3 with introducing *stochastic satisfiability modulo the theories of non-linear arithmetic and ODEs* as the unification of stochastic propositional satisfiability [47] and satisfiability modulo non-linear arithmetic [32] and modulo ODEs [29]. Section 4 formalizes the SSMT encoding of the probabilistic bounded reachability properties of PHA. Solving this symbolic encoding by an extension of the SSMT algorithm from [61] to incorporate ODEs, which is explained in Section 5, we obtain a technique for analysis of probabilistic bounded reachability problems of probabilistic hybrid automata. This technique does neither resort to approximation by intermediate finite-state abstractions of the hybrid system nor does it require flattening out concurrent system by computing an automaton-product, as the handling of discrete states is fully symbolic. The latter is demonstrated in Section 6, where a system with 13 parallel components is analyzed which would flatten out to more than a million discrete states.

With its focus on scalability in the number of concurrent components and in the size of the discrete components, being able to analyze e.g. networked automation systems at the level of their feedback dynamics with the continuous domain, we consider this line of work complementary to the body of analytic techniques developed for classes of hybrid systems with more intricate stochastic dynamics. Our approach is very confined concerning the stochastic behavior; actually, it only admits probabilistic events within transitions. Analytic techniques based on Lyapunov theory (e.g. [30]) can handle eventual stabilization of switched stochastic differential equations and reachability theory of piecewise deterministic Markov processes [24] can characterize the reach sets of systems incorporating (deterministic) differential equations paired with

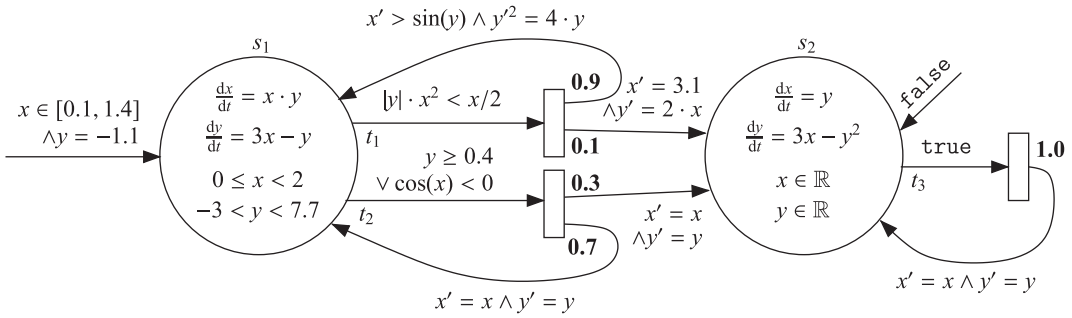


**Fig. 1**. A probabilistic hybrid automaton *A*. The rectangular boxes denote random events selecting a transition variant with the probabilities denoted along the transition arcs. Note that transition selection and execution is a three-stage process: Non-determinism, which may arise due to overlapping guard conditions, is resolved strictly before performing the random experiment selecting a transition variant determining the successor location. Hereafter, a continuous successor state is selected.

probabilistic switching governed by guards and state-dependent jump rates. Automation of these approaches is, however, hard and probably bound to be of confined scalability, as it requires techniques like the construction of a common Lyapunov function, computation of simulating systems, or the conservative, yet sufficiently accurate approximation of, e.g. hitting distributions [20], which using usual approximation schemes for distributions is exponential in the continuous dimension of the system and subject to state explosion in the discrete states. It would be interesting to see how these approaches could be unified, overcoming the limitations of our current approach concerning the classes of continuous behavior while retaining its scalability on concurrent systems.

## 2. Dense-time probabilistic hybrid automata

Probabilistic hybrid automata (PHA) extend the concept of hybrid automata with probabilistic transitions modeling, e.g. component failures or collisions on a shared medium and the resulting loss of data packets in distributed systems. Like dense-time hybrid automata, PHA feature a finite set of discrete locations or modes, each of which comes decorated with a (possibly non-linear) differential equation governing the dynamics of a vector of continuous variables while residing in the mode and with an invariant constraining the evolution of the continuous variables while in the mode. Mode changes are effected by instantaneous transitions guarded by conditions on the current values of the continuous variables, and such mode changes may also involve discontinuous updates to the continuous variables. Both transition selection and variable updates may be non-deterministic; the first situation arises in case of overlapping guard conditions, the second due to underspecification in the pre–post-relations defining these assignments. Beyond these mechanisms from hybrid automata, PHA add the probabilistic selection of a transition variant based on a random experiment: following the idea of Sproston [59,60], potentially non-deterministic transition selection happens first, choosing one transition among the enabled ones, and is then followed by randomized selection of a transition variant according to a probability distribution belonging to the selected transition. The different transition variants can lead to different follow-up locations and different continuous successors, as depicted in Fig. 1, where the guard condition determining transition selection is depicted to the left of the box denoting the random experiment. In comparison to Piecewise Deterministic Markov Processes [24], PHA add a second form of choice dynamics, namely non-deterministic choice in both the selection among competing transitions and the computation of the continuous successor state, yet omit transition rates. The rationale for adding non-determinism is enhanced expressiveness when addressing open dynamic systems or partially developed technical systems, where part of the uncertain system dynamics, like component failures in the subsystem modelled, can be characterized statistically, while other dynamical aspects are unknowns at the time of modeling and analysis and thus need to be accounted for in their worst-case form, as permitted by non-deterministic underspecification under a demonic interpretation of non-determinism.

Formally, a (flat, we will introduce concurrency later on) *dense-time probabilistic hybrid automaton* $A = (\Lambda, \mathit{Trans}, R, s, p, g, \mathit{asgn}, \mathit{ode}, \mathit{inv}, \mathit{init})$ consists of the following components:

- Finite sets $\Lambda$ of *locations*, *Trans* of *transitions*, and $R = \{x_1, \ldots, x_n\}$ of *continuous state components*, together with mappings $s : \mathit{Trans} \xrightarrow{\text{total}} \Lambda$, assigning to each transition its source location, and $p : \mathit{Trans} \xrightarrow{\text{total}} P(\Lambda)$, assigning to each transition a probability distribution over the target locations. [1] Here and in the sequel, $P(M)$ denotes the set of probability distributions over the finite set $M$.
- A family $g = (g_t)_{t \in \mathit{Trans}}$ assigning to each transition a *transition guard* enabling that transition, where the transition guard is an arithmetic predicate with free variables in $R$ that may involve non-linear arithmetic and transcendental functions, as can be seen in Fig. 1.

[1] W.l.o.g., distributions range over the full set $\Lambda$ as unconnected locations and locations connected with probability 0 are indistinguishable w.r.t. probabilistic reachability.

- A family $asgn = (asgn_{t,\lambda'})_{t\in Trans,\lambda'\in\Lambda}$ assigning to each transition and each target location an *assignment* which is defined by means of a predicate over variables in $R$ and $R'$, where $R' = \{x'_1, \ldots, x'_n\}$ denotes primed variants of the state components in $R$. Undecorated state components $x \in R$ refer to the state immediately before the transition, while the primed variant $x' \in R'$ refers to the state immediately thereafter such that the predicates define pre–post-relations. The assignment predicates are potentially non-linear arithmetic predicates over the continuous variables and may involve transcendental functions; Fig. 1 provides examples of such assignment predicates.

  To maintain the desired separation between the resolution of non-determinism and random transitions, we demand that assignments are defined for each state satisfying the guard, i.e. require $g_t \Rightarrow \exists x'_1, \ldots, x'_n : asgn_{t,\lambda'}$ to be valid for each $t \in Trans$ and each $\lambda' \in \Lambda$ with $p(t)(\lambda') > 0$.
- A family $ode = (ode_\lambda)_{\lambda\in\Lambda}$ assigning to each location $\lambda \in \Lambda$ a *flow* $ode_\lambda : \mathbb{R}^n \xrightarrow{\text{total}} \mathbb{R}^n$ which describes the continuous evolution while residing in $\lambda$ by means of a vector field, constraining the evolution to solutions of the ordinary differential equation $\frac{d\vec{x}}{dt} = ode_\lambda(\vec{x})$. For technical reasons induced by the constraint solving mechanisms for ODEs, we assume in the sequel that the automaton follows each individual flow for a total duration of at most a given $\Delta > 0$, thereafter being forced into a stutter step before resuming the flow. Note that due to stuttering, this assumption does not prevent the automaton from residing in $\lambda$ for an arbitrarily long duration, to the extent permitted by the invariant.
- A family $inv = (inv_\lambda)_{\lambda\in\Lambda}$ assigning to each location $\lambda \in \Lambda$ an *invariant* $inv_\lambda$ which is a box in $\mathbb{R}^n$, i.e. specifies for each $x_i \in R$ an interval $I_{x_i}$ of finite width that the continuous evolution may not leave while residing in $\lambda$.
- A family $init = (init_\lambda)_{\lambda\in\Lambda}$ of *initial state predicates*, where each $init_\lambda$ is an arithmetic predicate over $R$ which constrains the valuations of the continuous state components when control resides *initially* in the discrete location $\lambda$.[2]

The automaton engages in a sequence of continuous flows and discrete jumps, where the continuous flows are solutions to the ordinary differential equations assigned to the current location and the discrete jumps coincide with enabled transitions, thereby first selecting non-deterministically among the enabled transitions and then probabilistically among the different target locations and finally again non-deterministically among the assignments to continuous variables permitted by the transition.

Given a transition $tr \in Trans$, $A$ has a *tr-jump* from state $(\lambda, \vec{x}) \in \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$ to state $(\lambda', \vec{x}') \in \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$ if $s(tr) = \lambda$ and $\vec{x} \models g_{tr}$ and if $\vec{x}, \vec{x}' \models asgn_{tr,\lambda'}$ or $p(tr)(\lambda') = 0$.[3] Here, $\vec{x}, \vec{x}' \models asgn_{tr,\lambda'}$ denotes that $asgn_{tr,\lambda'}$ is satisfied when $\vec{x}$ is substituted for the variables in $R$ and $\vec{x}'$ is substituted for the variables in $R'$ or alternatively . Reflecting the random event of selecting a target location entailed in the transition, the *probability of the tr-jump* from $(\lambda, \vec{x})$ to $(\lambda', \vec{x}')$ is $pr = p(tr)(\lambda')$. In this case, we write $(\lambda, \vec{x}) \to^{pr}_{tr} (\lambda', \vec{x}')$.

$A$ has a continuous *flow* from $(\lambda, \vec{x}) \in \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$ to $(\lambda', \vec{x}') \in \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$ if $\lambda = \lambda'$ and if there is a continuous evolution in location $\lambda$ of duration at most $\Delta$ which leads from $\vec{x}$ to $\vec{x}'$, i.e. if there is a duration $t \in ]0, \Delta]$ such that

$$\exists F : [0,t] \xrightarrow{\mathcal{C}^1} \mathbb{R}^n : \begin{pmatrix} F(0) = \vec{x} \\ \wedge\ F(t) = \vec{x}' \\ \wedge\ \forall \delta \in [0,t] : \frac{dF}{dt}(\delta) = ode_\lambda(F(\delta)) \\ \wedge\ \forall \delta \in [0,t] : F(\delta) \in inv_{s(tr)} \end{pmatrix}$$

where $A \xrightarrow{\mathcal{C}^1} B$ denotes the continuously differentiable functions with domain $A$ and image $\subseteq B$. In this case, we write $(\lambda, \vec{x}) \to_t (\lambda', \vec{x}')$.

A *run* of $A$ is a sequence $(\lambda_0, \vec{x}_0) \to_{t_1} (\lambda_1, \vec{x}_1) \to^{pr_2}_{tr_2} \ldots (\lambda_n, \vec{x}_n)$ of flows and jumps. It need not alternate between flows and jumps, but may well chain multiple jumps or multiple flows in a row, thus supporting stuttering within flows as well as permitting multiple jumps at the same time instant. The probability $Pr(r)$ of a run $r$ is the product of the probabilities of the jumps incorporated, i.e. for the above run it is $\prod_{i=1}^{n} pr_i$ with $pr_i = 1$ whenever the step is a continuous evolution and the probability of the corresponding jump otherwise. Given a run, we call its projection to the states visited, i.e. the sequence $\langle(\lambda_0, \vec{x}_0), (\lambda_1, \vec{x}_1), \ldots, (\lambda_n, \vec{x}_n)\rangle$ its *trace*.

Note that beyond the stochastic uncertainty of the PHA dynamics, which is covered by the random events in transitions, there are two points in a step where uncertainty modelled as non-determinism enters. These are in the selection among the set of enabled transitions, which need not be a singleton, and among the continuous successor state in the assignment to the continuous variables. As usual in models blending probabilistic and non-deterministic choice, like Markov Decision Processes [10], we assume that the dynamics is controlled by a *decision maker* or *policy (scheduler, adversary)* resolving the

[2] A discrete location $\lambda$ not to be taken initially takes the predicate $init_\lambda = \texttt{false}$.

[3] The alternative $p(tr)(\lambda') = 0$ completes the transition system with default transitions of probability 0 in cases where no explicit transition is available, as arcs with probability 0 are generally not drawn in PHA and thus naturally associated with the unsiatisfiable transition predicate `false`. This completion is just a technicality which avoids many case distinctions in the subsequent development.

non-determinism based on (complete) observation of the current state and history. Based on the trajectory exhibited so far, such a policy can decide

1. whether a jump shall be performed, if one is enabled, or whether and for how long the current continuous evolution shall proceed,
2. in case a jump is performed, which of the enabled transitions $tr \in Trans$ shall be taken, and
3. depending on the probabilistically generated random event, which of the possible updates of continuous variables pertaining to this event shall be performed.

In the remainder, we shall assume that these decisions depend deterministically on the current trajectory prefix, i.e. that the permissible adversaries are Markovian deterministic policies [6], also known as step-dependent schedulers. Consequently, a policy consists of

1. a start state selected amongst the permissible initial states $\{(\lambda, \vec{x}) \mid \vec{x} \models init_\lambda\}$ and
2. a mapping from sequences of sampling points to either a duration $t \leq \Delta$ of the next flow or a transition $tr \in Trans$ to be taken plus a $\Lambda$-indexed family of assignments to the continuous variables (one assignment for each outcome of the random event, thus covering the reaction of the policy to the random event).

I.e. with $S = \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$ being the state set of the PHA, a policy is a pair of an initial state and a mapping $Pol : S^* \xrightarrow{\text{total}} \left(]0, \delta] \cup \left(Trans \times (\Lambda \xrightarrow{\text{total}} (R \xrightarrow{\text{total}} \mathbb{R}))\right)\right)$ with the side condition that the moves suggested by the policy are feasible in the PHA. The latter amounts to demanding that whenever $(\lambda, \vec{x}) \in S$ is the last element of the sequence $s \in S^*$ then

$$\exists(\lambda', \vec{x}') \in S : (\lambda, \vec{x}) \to_t (\lambda', \vec{x}') \text{ if } Pol(s) = t \in ]0, \Delta]$$

and

$$\forall \lambda' \in \Lambda : (\lambda, \vec{x}) \to_{tr}^{p(tr)(\lambda')} (\lambda', pa(\lambda')) \text{ if } Pol(s) = (tr, pa)$$

Given a set of target states $Target = (Target_\lambda)_{\lambda \in \Lambda}$, a depth bound $k \in \mathbb{N}$, and a policy $((\lambda, \vec{x}), Pol)$, the *depth-bounded reachability problem* for depth $k$ subject to policy $((\lambda, \vec{x}), Pol)$ can be defined as follows: first, a trace $\langle(\lambda_0, \vec{x}_0), (\lambda_1, \vec{x}_1), \ldots, (\lambda_k, \vec{x}_k)\rangle \in S^*$ is *consistent with the policy* if

1. it is anchored in the start state suggested by the policy, i.e. $(\lambda_0, \vec{x}_0) = (\lambda, \vec{x})$, and
2. the steps are policy-consistent, i.e. for all $i < k$ we have that $(\lambda_i, \vec{x}_i) \to_t (\lambda_{i+1}, \vec{x}_{i+1})$ if $Pol(\langle(\lambda_0, \vec{x}_0), \ldots, (\lambda_i, \vec{x}_i)\rangle) = t$ and that $(\lambda_i, \vec{x}_i) \to_{tr}^{pr} (\lambda_{i+1}, \vec{x}_{i+1})$ and $\vec{x}_{i+1} = pa(\lambda_{i+1}))$ if $Pol(\langle(\lambda_0, \vec{x}_0), \ldots, (\lambda_i, \vec{x}_i)\rangle) = (tr, pa)$.

The trace $\langle(\lambda_0, \vec{x}_0), (\lambda_1, \vec{x}_1), \ldots, (\lambda_k, \vec{x}_k)\rangle \in S^*$ *hits the target* if $\exists i \leq k : \vec{x}_i \models Target_{\lambda_i}$. The *probability of hitting the target within k steps under policy* $((\lambda, \vec{x}), Pol)$ then is

$$Pr^k_{((\lambda,\vec{x}),Pol)} = \sum_{\{s \in S^{k+1} \mid s \text{ consistent with } ((\lambda,\vec{x}),Pol), s \text{ hits target}\}} Pr(s)$$

where $Pr(s)$ is the probability of sequence $s$, which is defined as the probability $Pr(r)$ of the (unique) policy-consistent run $r$ yielding trace $s$. It follows from a straightforward induction on the depth-bound $k$ that for any policy, this probability of reaching *Target* under the given policy can be characterized recursively as follows:

**Lemma 1.** *Given a PHA* $A = (\Lambda, Trans, R, s, p, g, asgn, ode, inv, init)$ *and a corresponding policy* $((\lambda, \vec{x}), Pol)$*, the probability of hitting the target within k steps under the policy satisfies*

$$Pr^k_{((\lambda,\vec{x}),Pol)} = \begin{cases} 1 & \text{if } \vec{x} \models Target_\lambda, \\ 0 & \text{if } k = 0 \wedge \vec{x} \not\models Target_\lambda, \\ Pr^{k-1}_{((\lambda',\vec{x}'),Pol)} & \text{if } k > 0 \wedge Pol(\lambda, \vec{x}) = t \wedge \\ & \quad (\lambda, \vec{x}) \to_t (\lambda', \vec{x}'), \\ \sum_{\lambda' \in \Lambda} p_{tr}(\lambda') \cdot Pr^{k-1}_{((\lambda', pa(\lambda')),Pol)} & \text{if } k > 0 \wedge Pol(\lambda, \vec{x}) = (tr, pa) \end{cases}$$

In the sequel, we will be interested in the *maximum probability under an arbitrary policy* of reaching a given set of target locations within a given number $k \in \mathbb{N}$ of steps. Semantically, this is adequate for modelling and analyzing situations where the target states are considered undesirable and a demonic perspective to non-determinism is taken (rendering the policy

adversarial), or symmetrically to cases where the target states are considered desirable and an angelic perspective (rendering the policy cooperative) is taken. In particular, depth-bounded probabilistic reachability in probabilistic hybrid systems is representative of a number of verification problems for embedded systems, e.g.

- performing quantitative safety analysis (in the sense of estimating failure probability) of a conflict resolution scheme which is expected to terminate after a finite number of actions whenever triggered, like collision avoidance maneuvers in road traffic,
- performing quantitative safety analysis (in the sense of estimating failure probability) of a finite critical mission, like the descent of an airplane,
- assessing the reliability of a system subject to regular maintenance, where the number of system actions between maintenance is bounded by a constant $k$, or
- step-bounded region stabilization [4] of hybrid systems subject to probabilistic disturbances, i.e. determining whether a system will with sufficient probability converge into a target region within a given step (and thus, time) bound.

We will elaborate on how to formalize these proof obligations after presenting a symbolic encoding of the probabilistic hybrid system in Section 4.

Formally, the *maximum probability of hitting the target within k steps* is defined as the maximum over arbitrary policies of the policy-dependent hit probability, i.e. as probability $Pr^k = \max_{((\lambda,\vec{x}),Pol) \text{ a policy}} Pr^k_{((\lambda,\vec{x}),Pol)}$. The *probabilistic bounded model checking problem (PBMC, for short)* is then defined to be the problem of deciding whether the maximum probability of reaching the undesirable states within a given number of steps is below a given threshold:

**Definition 1** (*Probabilistic bounded model checking*)**.** Given a probabilistic hybrid automaton $A = (\Lambda, Trans, R, s, p, g, asgn, ode, inv, init)$ together with a (predicatively defined) set of target states $Target = (Target_\lambda)_{\lambda\in\Lambda}$, a depth $k \in \mathbb{N}$, and a probability threshold $\theta \in [0, 1]$, the *probabilistic bounded model checking problem w.r.t. target states Target and depth k* is to determine whether $Pr^k < \theta$.

The maximum probability of hitting the target within $k$ steps can be computed by a backward induction scheme akin to that of Bellman [10], yielding the following inductive characterization of the probability of reaching a target state.

**Lemma 2** (Probabilistic bounded reachability)**.** *Given a probabilistic hybrid automaton* $A = (\Lambda, Trans, R, s, p, g, asgn, ode, inv, init)$ *together with a set of target states* $Target = (Target_\lambda)_{\lambda\in\Lambda}$*, a depth bound* $k \in \mathbb{N}$*, and a hybrid state* $(\lambda, \vec{x}) \in \Lambda \times (R \xrightarrow{\text{total}} \mathbb{R})$*, the maximum probability of reaching the target Target within at most k steps from the initial state* $(\lambda, \vec{x})$ *– i.e. under any policy selecting* $(\lambda, \vec{x})$ *as its initial state –, denoted* $P^k_A(\lambda, \vec{x}, Target)$*, is*

$$P^k_A(\lambda, \vec{x}, Target) = \begin{cases} 1 & \text{if } \vec{x} \models Target_\lambda \\ 0 & \text{if } \vec{x} \not\models Target_\lambda \wedge k = 0 \\ \max(idealflow, idealjump) & \text{if } \vec{x} \not\models Target_\lambda \wedge k > 0 \end{cases}$$

*where* $idealflow = \max\{P^{k-1}_A(\lambda', \vec{x}', Target) \mid t \in ]0, \Delta], (\lambda, \vec{x}) \to_t (\lambda', \vec{x}')\}$ *and*
$idealjump = \max_{tr\in Trans} \sum_{\lambda'\in\Lambda} \left( p_{tr}(\lambda') \cdot \max\{P^{k-1}_A(\lambda', \vec{x}', Target) \mid (\lambda, \vec{x}) \to^{p_{tr}(\lambda')}_{tr} (\lambda', \vec{x}')\} \right)$.

**Proof.** By induction over the depth-bound $k$ of the transition tree:

In case $k = 0$, the set of traces of length $k$ starting in $(\lambda, \vec{x})$ consists of exactly one trace $s$ comprising just the start state $(\lambda, \vec{x})$, i.e. $s = \langle(\lambda, \vec{x})\rangle$. This trace $s$ hits the target iff $\vec{x} \models Target_\lambda$. For each policy selecting $(\lambda, \vec{x})$ as initial state, $s$ is the unique policy-consistent trace of length 0, and thus has probability 1. Consequently,

$$P^0_A(\lambda, \vec{x}, Target) = \begin{cases} 1 & \text{if } \vec{x} \models Target_\lambda \\ 0 & \text{if } \vec{x} \not\models Target_\lambda \end{cases}$$

In case $k > 0$, we can build on the induction hypothesis that for each $(\lambda', \vec{x}')$, the maximum probability under any scheduler of reaching *Target* within $k-1$ steps from $(\lambda', \vec{x}')$ is $P^{k-1}_A(\lambda', \vec{x}', Target)$ and that there exist a policy $((\lambda', \vec{x}'), Pol_{\lambda',\vec{x}',k-1})$ starting in $(\lambda', \vec{x}')$ and providing exactly that reach probability within $k - 1$ steps.

[4] Note that eventual stabilization, while frequently considered due to its simpler mathematics, is hardly ever a convincing notion in practice. In most practical applications, bounds on stabilization time are desirable.

Now, we define a policy $((\lambda, \vec{x}), Pol)$ for the $k$-step case by

$$Pol(\langle(\lambda_1, \vec{x}_1), \ldots, (\lambda_n, \vec{x}_n)\rangle) = \begin{cases} Pol_{\lambda_2, \vec{x}_2, k-1}(\langle(\lambda_2, \vec{x}_2), \ldots, (\lambda_n, \vec{x}_n)\rangle) & \text{if } n > 1 \\ t_{ideal} & \text{if } n = 1 \text{ and } idealflow \geq idealjump \\ (tr, pa)_{ideal} & \text{if } n = 1 \text{ and } idealflow < idealjump \end{cases}$$

where *idealflow* and *idealjump* are defined as before and $t_{ideal}$ satisfies $(\lambda_1, \vec{x}_1) \rightarrow_{t_{ideal}} (\lambda', \vec{x}') \wedge P_A^{k-1}(\lambda', \vec{x}', Target) = idealflow$, i.e. is the flow duration leading to the flow successor with maximal $P_A^{k-1}(\lambda', \vec{x}', Target)$. Likewise, $(tr, pa)_{ideal}$ is the jump with $\sum_{\lambda' \in \Lambda} \left( p_{tr}(\lambda') \cdot \max\{P_A^{k-1}(\lambda', \vec{x}', Target) \mid (\lambda_1, \vec{x}_1) \rightarrow_{tr}^{p_{tr}(\lambda')} (\lambda', \vec{x}')\} \right) = idealjump$, i.e. selects the transition and assignments with the maximum weighted sum of the successor's $P_A^{k-1}(\lambda', \vec{x}', Target)$.

It follows from Lemma 1 that $A$ under policy $((\lambda, \vec{x}), Pol)$ reaches *Target* with probability $P_A^k(\lambda, \vec{x}, Target)$. It remains to be shown that there is no policy yielding a higher probability of reaching the target. Therefore, let $(((\lambda, \vec{x}), Pol')$ be an arbitrary policy starting in $(\lambda, \vec{x})$. Concerning the behavior of $Pol'$ on $(\lambda, \vec{x})$, we can distinguish two cases: either, it takes a flow or a jump, i.e. $Pol'(\langle(\lambda, \vec{x})\rangle) = t^*$ for some $t > 0$ or $Pol(\langle(\lambda, \vec{x})\rangle) = (tr^*, pa^*)$.

In case 1, i.e. if $Pol'$ selects a flow duration $Pol'(\langle(\lambda, \vec{x})\rangle) = t^*$, there is a state $(\lambda^*, \vec{x}^*)$ such that $(\lambda, \vec{x}) \rightarrow_{t^*} (\lambda^*, \vec{x}^*)$. Then

$$\begin{aligned} & P_A^k(\lambda, \vec{x}, Target) \\ &= \max(idealflow, idealjump) \\ &\geq idealflow \\ &= \max\{P_A^{k-1}(\lambda', \vec{x}', Target) \mid t \in ]0, \Delta], (\lambda, \vec{x}) \rightarrow_t (\lambda', \vec{x}')\} \\ &\geq P_A^{k-1}(\lambda^*, \vec{x}^*, Target) \\ &\geq Pr^{k-1}_{((\lambda^*, \vec{x}^*), Pol')} \\ &= Pr^{k}_{((\lambda, \vec{x}), Pol')} \end{aligned}$$

where the last inequation follows from the induction hypothesis and the last equation from Lemma 1. This shows that $Pol'$ yields at most the same probability of reaching *Target*.

Case 2, i.e. that $Pol'$ takes a jump, is similar . □

## 3. Stochastic satisfiability modulo theories

The *stochastic satisfiability modulo theories* (SSMT) problem was introduced in [33,61] as an extension of the *satisfiability modulo theories* (SMT) problem (e.g. [7]) to support *existential* and *randomized quantification* over discrete variables as known from *stochastic satisfiability* [SSAT, 47,56] and *stochastic constraint satisfaction* [SCSP, 63]. In contrast to SSAT and SCSP problems, where the domains of all variables are finite, the original formulation of SSMT also permits continuous domains for non-quantified variables (which are interpreted as the set of innermost existentially quantified variables), and involves non-linear real arithmetic including transcendental functions like sin and exp. In this paper, we enhance the expressive power of SSMT beyond [33,61] by two major extensions:

1. Integration of *ordinary differential equations* (ODEs).
2. Existential quantification over *real-valued* variables.

The integration of ODEs and the possibility to quantify over real-valued variables enhances the SSMT paradigm such that it can represent the semantics of *dense-time* hybrid systems.

### 3.1. Syntax of SSMT

A stochastic satisfiability modulo theory problem

$$\Phi = Q_1 x_1 \in \mathrm{dom}(x_1) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi$$

is specified by a *prefix* $Q_1 x_1 \in \mathrm{dom}(x_1) \ldots Q_n x_n \in \mathrm{dom}(x_n)$ binding the variables $x_i$ to the quantifiers $Q_i \in \{\exists, \mathfrak{R}_d\}$, and a quantifier-free SMT formula $\varphi$, also called the *matrix*. A quantifier $Q$, associated with variable $x$, is either *existential*, denoted as $\exists$, or *randomized*, denoted as $\mathfrak{R}_d$, where $d$ is a discrete probability distribution over the finite domain of $x$. In this paper,

we do not require that the domain $\mathrm{dom}(x)$ of an existentially quantified variable $x$ has to be finite but can also be a *bounded real-valued interval*.

Intuitively, the value of a variable $x$ bound by a randomized quantifier (*randomized variable* for short) is determined stochastically by the corresponding distribution $d$, while the value of an existentially quantified variable (*existential variable* for short) can be set arbitrarily. We usually denote such a probability distribution $d$ by a function $[v_1 \to p_1, \ldots, v_m \to p_m]$ which maps value $v_i$ to a probability $p_i$ with $0 < p_i \leq 1$. The mapping $v_i \to p_i$ is understood as $p_i$ is the probability of setting variable $x$ to value $v_i$. The distribution satisfies $v_i \neq v_j$ for $i \neq j$, $\sum_{i=1}^{m} p_i = 1$, and $\mathrm{dom}(x) = \{v_1, \ldots, v_m\}$. For instance, $\mathfrak{R}_{[0\to 0.2, 1\to 0.5, 2\to 0.3]} x \in \{0, 1, 2\}$ means that the variable $x$ is assigned the value 0, 1, or 2 with probability 0.2, 0.5, and 0.3, respectively.

Let $V(\varphi)$ be the set of all variables occurring in an SMT formula $\varphi$. To cope also with ODE constraints, $\varphi$ contains two classes of variables: first, variables of base types real and integer, denoted $V_{base}(\varphi) \subseteq V(\varphi)$, and second, ODE variables $V_{ODE}(\varphi) \subseteq V(\varphi)$ (cf. Section 3.2.2). We require that only base-type variables are quantified and that $\Phi$ has no free base-type variables, i.e. $V_{base}(\varphi) = \{x_1, \ldots, x_n\}$, where $\{x_1, \ldots, x_n\}$ is the set of variables bound in the quantifier prefix. Without loss of generality, the SMT formula $\varphi$ is in conjunctive normal form (CNF), i.e. $\varphi$ is a conjunction of clauses where a clause is a disjunction of constraints. A constraint can be either a (non-linear) arithmetic constraint like $\sin(x^2)/y \geq \exp(x \cdot y)$ or an ODE constraint like $\left(\frac{dz_1}{dt}(t) = z_1 \cdot sin(z_2), \mathtt{true}, x, y, \tau\right)$. Formally, the syntax of an SMT formula $\varphi$ is as follows.

$$
\begin{aligned}
\mathit{smt\_formula} &::= \{\mathit{clause} \wedge\}^* \mathit{clause} \\
\mathit{clause} &::= (\{\mathit{constraint} \vee\}^* \mathit{constraint}) \\
\mathit{constraint} &::= \mathit{arithmetic\_constraint} \mid \mathit{ode\_constraint} \\
\mathit{arithmetic\_constraint} &::= \mathit{term}\ \mathit{relop}\ \mathit{term} \\
\mathit{ode\_constraint} &::= (d\ \mathit{ode\_var}/dt = \mathit{oterm}, \\
&\qquad \mathit{invar}, \mathit{base\_var}, \mathit{base\_var}, \mathit{base\_var}) \\
\mathit{invar} &::= \mathtt{true} \mid \mathit{const}\ (< \mid \leq)\ \mathit{ode\_var}\ (< \mid \leq)\ \mathit{const} \\
\mathit{term} &::= \mathit{uop}\ \ \mathit{term} \mid \ \mathit{term}\ \mathit{bop}\ \ \mathit{term} \mid \mathit{base\_var} \mid \mathit{const} \\
\mathit{oterm} &::= \mathit{uop}\ \mathit{oterm} \mid \mathit{oterm}\ \mathit{bop}\ \mathit{oterm} \mid \mathit{ode\_var} \mid \mathit{const} \\
\mathit{relop} &::= < \mid \leq \mid = \mid \geq \mid > \\
\mathit{uop} &::= \sin \mid \exp \mid \mathrm{sqrt} \mid \ldots \\
\mathit{bop} &::= + \mid - \mid \cdot \mid \ldots
\end{aligned}
$$

where *base_var* and *ode_var* are base-type and ODE variables, respectively, and *const* ranges over the rational constants.

As an example consider the following SSMT formula.

$$
\overbrace{\mathfrak{R}_{[1\to 0.3, 2\to 0.7]} a \in \{1, 2\}\ \exists x \in [-1, 3]\ \exists y \in [2, 5]\ \exists \tau \in [0, 10]}^{\text{quantifier prefix}}\ :
$$

$$
\underbrace{\begin{aligned}
&\left(a = 1 \vee \left(\tfrac{dz}{dt}(t) = 1.4 \cdot z, \mathtt{true}, x, y, \tau\right)\right) \\
\wedge\ &\left(a = 2 \vee \left(\tfrac{dz}{dt}(t) = z^2 - 1.2 \cdot z, 2.5 \leq z \leq 7.1, x, y, \tau\right)\right) \\
\wedge\ &\left(y - x \geq 4.2\right)
\end{aligned}}_{\text{matrix}}
$$

### 3.2. Semantics of SMT

Before we formally introduce the semantics of SSMT, we need to define the satisfaction of quantifier-free SMT formulae in our context. Let $\varphi$ be an SMT formula including non-linear arithmetic constraints and ODE constraints.[5] Then, we call $\sigma$ a *total valuation* if it maps each base variable $x$ in $V_{base}(\varphi)$ to a value in $\mathbb{R}$ and each ODE variable from $V_{ODE}$ to a function, respectively. More precisely, for base-type variables $x \in V_{base}$ we have $\sigma : x \mapsto \sigma(x) \in \mathbb{R}$, and for ODE variables $x \in V_{ODE}$, $\sigma : x \mapsto \sigma(x) \in \mathcal{C}^1_{fin}$, where $\mathcal{C}^1_{fin}$ denotes the set of differentiable functions on an interval, i.e. $\mathcal{C}^1_{fin} := \left\{x \in \mathcal{C}^1 | dom(x) = [0, a], a \geq 0\right\}$. An example may help to clarify this definition of $\sigma$. For two variables $a, b \in V_{base}$ and a variable $c \in V_{ODE}$, a possible valuation $\sigma$ could be given by $\sigma(a) = 3.2$, $\sigma(b) = -2.1$ and $\sigma(c) = t \mapsto c(t) = 2.3 \cdot t^2$ with $t \in [0, 10]$. The base variables are

[5] ODE constraints and ODE variables will be introduced in more detail in Sect. 3.2.2. For the moment, considering them as constraints and variables of a special type is sufficient.

thereby mapped to values from $\mathbb{R}$, while the ODE variable $c$ is mapped to a function $[0, 10] \mapsto \mathbb{R}$ which is differentiable over its domain. A valuation can thus be interpreted as a function $\sigma : (V_{base} \cup V_{ODE}) \to (\mathbb{R} \cup \mathcal{C}^1_{fin})$, though above definition is more precise about which elements are mapped to real numbers and which to functions. There can obviously be an infinite number of valuations for these three variables – and this observation is true in general. The following rules on *satisfaction of constraints* will set conditions on $\sigma$ that can be interpreted as a filter removing those valuations that do not actually constitute satisfying valuations and are therefore of little interest.

*3.2.1. Arithmetic constraints*

Satisfaction of arithmetic constraints is w.r.t. the standard interpretation of the arithmetic operators and the ordering relations over the reals. For instance, $x^2 > \sin(y)$ is satisfied under the valuation $\sigma$ with $\sigma(x) = -5.12$ and $\sigma(y) = 3.8$ because $(-5.12)^2 > \sin(3.8)$.

*3.2.2. ODE constraints*

To describe the continuous evolutions of variables, we consider the integration of the theory of *ordinary differential equations* (ODEs) into SSMT in addition to non-linear arithmetic over the reals and integers.

**Definitions** We call the tuple $c_{ODE} := (ODE(x), I(x), u, v, \tau)$ an *ODE constraint*, with

- a differential equation $ODE(x) := \frac{dx}{dt}(t) = f(x_1(t), \ldots, x_n(t))$ whose right hand side function $f : \mathbb{R}^n \to \mathbb{R}$ is Lipschitz continuous on the given domain $dom(x_1) \times \cdots \times dom(x_n)$, where $x_1, \ldots, x_n \in V_{ODE}(\varphi)$ are the variables defined by this or other ODE constraints (see below),
- an invariant $I(x) := lb \leq x(t) \leq ub$, i.e. a constraint describing an interval,
- a variable $u$ called the start point of the trajectory in dimension $x$,
- a variable $v$ called the end point of the trajectory in dimension $x$, and
- a variable $\tau$ called the length of the trajectory.

A variable $x$ is *defined by* an ODE constraint, *ODE variable* for short, iff it occurs on the left-hand side of the differential equation $ODE(x)$ of an ODE constraint $c_{ODE}$.

**Semantics of ODE constraints** An ODE constraint $c = (ODE(x_i), I(x_i), u_i, v_i, \tau_i)$ is satisfied by a valuation $\sigma$ if it contains solution functions $\sigma(x_1), \ldots, \sigma(x_n)$ such that $\sigma(x_i)$ connects the valuations given for the start point $\sigma(u_i)$ with the valuations for the end point $\sigma(v_i)$ by a trajectory of length $\sigma(\tau_i)$, the solution functions satisfy the differential equation $ODE(x_i)$, and $\sigma(x_i)$ never leaves the invariant $I(x_i)$. We thus call $c_{ODE}$ satisfied by a valuation $\sigma$ iff the above properties are satisfied. More formally:

$$eval((ODE(x_i), I(x_i), u_i, v_i, \tau_i), \sigma)$$
$$= \begin{cases} \texttt{true} & \text{if } \max(\mathrm{dom}(\sigma(x_i))) = \sigma(\tau_i) \\ & \text{and } \frac{d(\sigma(x_i))}{dt}(t) = f(\sigma(x_1)(t), \ldots, \sigma(x_n)(t)) \text{ for all } t \in [0, \sigma(\tau_i)] \\ & \text{and } \sigma(x_i)(0) = \sigma(u_i) \text{ and } \sigma(x_i)(\sigma(\tau_i)) = \sigma(v_i) \\ & \text{and } lb \leq \sigma(x_i)(t) \leq ub \text{ for all } t \in [0, \sigma(\tau_i)] \\ \texttt{false} & \text{else} \end{cases}$$

Fig. 2 illustrates these characteristics of a satisfying valuation.

We call a set of ODE constraints $S$ satisfied under a valuation $\sigma$ iff all its elements are satisfied under $\sigma$. As above, $eval(S, \sigma)$ returns `true` iff $S$ is satisfied according to this definition.

*3.2.3. Satisfaction of SMT formulae*

Given an SMT formula $\varphi$, and a total valuation $\sigma$ of $V(\varphi)$, $\varphi$ is *satisfied under* $\sigma$ iff at least one constraint in each clause is satisfied under $\sigma$. Such a valuation $\sigma$ is called *solution of* $\varphi$. Let $\pi$ be a partial valuation of $V(\varphi)$. An SMT formula $\varphi$ is called *satisfiable under* $\pi$ iff there exists a solution $\sigma$ of $V(\varphi)$ s.t. for each variable $x$, if $\pi(x)$ is defined then $\sigma(x) = \pi(x)$ holds.

*3.3. Semantics of SSMT*

The semantics of an SSMT problem is defined by the *probability of satisfaction* [33,61]. Formally, the *probability of satisfaction* $Pr(\Phi)$ of an SSMT formula $\Phi = Q_1 x_1 \in \mathrm{dom}(x_1) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi$ is given by the value $prob(\Phi, \emptyset) \in [0, 1]$. Here, $\emptyset$ denotes the partial valuation which is totally undefined. Given a partial valuation $\pi$, the partial valuation $\pi' = \pi \oplus [x \mapsto v]$ is defined as $\pi'(x) = v$ and $\forall y \neq x : \pi'(y) = \pi(y)$. A valuation $\pi'$ is called an extension of $\pi$ if $\mathrm{dom}(\pi') \supset \mathrm{dom}(\pi)$

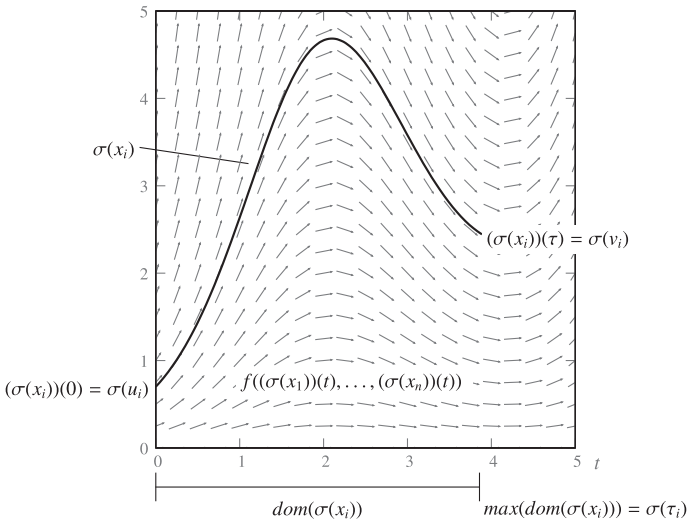


**Fig. 2**. A satisfying valuation for an ODE constraint.

and $\forall y \in \mathrm{dom}(\pi) : \pi'(y) = \pi(y)$. The function $prob(\Phi, \pi)$, where $\Phi$ is an SSMT formula, *Pre* is a quantifier prefix with $\varepsilon$ denoting the empty prefix, and $\pi$ is a partial valuation of $V(\varphi)$, is defined recursively by the following rules.

1. $prob(\varepsilon : \varphi, \pi) = 0$ if $\varphi$ is not satisfied by any extension of $\pi$.
2. $prob(\varepsilon : \varphi, \pi) = 1$ if $\varphi$ is satisfied by some extension of $\pi$.
3. $prob(\exists x \in \mathrm{dom}(x)\ Pre : \varphi, \pi) = \max_{v \in \mathrm{dom}(x)} prob(Pre : \varphi, \pi \oplus [x \mapsto v])$.
4. $prob(\mathfrak{R}_d x \in \mathrm{dom}(x)\ Pre : \varphi, \pi) = \sum_{v \in \mathrm{dom}(x)} d(v) \cdot prob(Pre : \varphi, \pi \oplus [x \mapsto v])$.

Note that $d$ is a probability distribution where $(v \to p) \in d$ denotes $d(v) = p$.

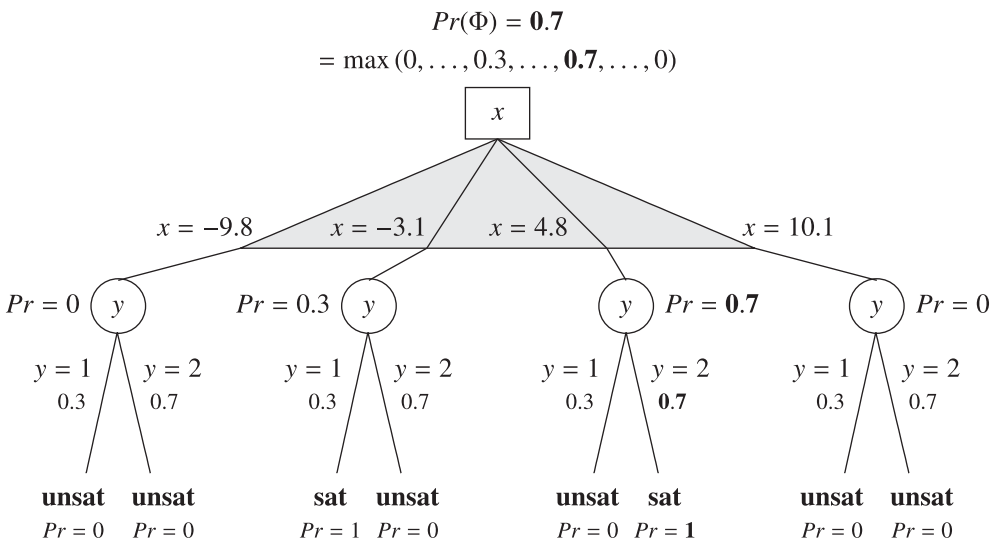


**Fig. 3**. Semantics of an SSMT formula depicted as a tree. Only a finite sample of the infinitely many branches departing from the grey area depicting the real-valued quantifier has been drawn.

The probability of satisfaction is determined by the function *prob* with the original SSMT formula $\Phi$ and the empty valuation $\emptyset$ as its inputs. Then, the quantifier prefix is traversed recursively from left to right, thereby substituting all possible values of the quantified variables. The goal for existential variables is to compute the maximum probability over all alternatives, while randomized quantification calls for calculating the weighted sum. Once the whole quantifier prefix is traversed, $\pi$ is a total valuation of all base-type variables in $V_{base}(\varphi)$ since we require $\Phi$ to feature no free variables in $V_{base}(\varphi)$. The satisfiability of $\varphi$ under $\pi$ then determines the probability of satisfaction for the base cases. For an example explaining this semantics confer Fig. 3.

Algorithmically, computing the probability of satisfaction for the extended notion of SSMT is not straightforward in general. This is first due to maximizing over uncountably many alternatives, and second due to dealing with undecidable theories as non-linear arithmetic and differential equations. To nevertheless find safe solutions to this important and challenging problem, we will present an algorithmic approach to effectively compute safe upper bounds of satisfaction probabilities in Section 5.

## 4. Reducing PBMC to SSMT

In order to perform probabilistic bounded model checking (PBMC) of dense-time probabilistic hybrid automata (PHA), we employ a reduction to stochastic satisfiability modulo theories (SSMT) which generalizes the propositional SAT encodings for bounded model checking of finite-state systems [16] and the SMT encodings for BMC of hybrid automata [5,31]. Our construction proceeds in two phases: first, we generate the matrix of the SSMT formula. This matrix is an SMT formula encoding all runs of $A$ of the given length $k \in \mathbb{N}$ which reached the goal states, akin to [5,31]. Thereafter, we add the quantifier prefix encoding the probabilistic and the non-deterministic choices, whereby a probabilistic choice reduces to a randomized quantifier while a non-deterministic choice yields an existential quantifier.

*Phase 1: Constructing the matrix*: Let $A = (\Lambda, \mathit{Trans}, R, s, p, \mathit{asgn}, \mathit{ode}, \mathit{inv}, \mathit{init})$ be a dense-time probabilistic hybrid automaton. In order to encode the runs of $A$ of some given length $k \in \mathbb{N}$ by a matrix formula, we proceed as follows:

*Reduction step 1.* For encoding the discrete states held during the steps of the hybrid systems, we take $k+1$ variables $\lambda^i$, for $0 \leq i \leq k$, each ranging over domain $\Lambda$. The value of $\lambda^i$ coincides with the discrete location which automaton $A$ resides in during step $i$.

*Reduction step 2.* For representing transitions connecting the above, we take $k$ variables $tr^i$ with domain *Trans*, for $1 \leq i \leq k$. The value of $tr^i$ encodes the $i$th move in the run of $A$.

*Reduction step 3.* For each continuous state component $x \in R$ we take $k+1$ real-valued variables $x^i$. The value of $x^{i-1}$ encodes the value of $x$ before the $i$th step in the run (and thus $x^i$ the value thereafter).

*Reduction step 4.* Furthermore, we take for each $x \in R$ a sequence of $k$ $\mathcal{C}^1$-valued variables $X^i$ representing the flows in steps $i = 1, \ldots, k$.

*Reduction step 5.* For representing the durations of continuous flows in the run, we take $k$ real-valued variables $t^i$ of range $[0, \Delta]$, one for each $1 \leq i \leq k$. The value of $t^i$ encodes the duration of the $i$th step. The step is a continuous flow if $t_i > 0$ and a jump otherwise.

*Reduction step 6.* The effect of the continuous flows on the state is encoded by the ODE constraint system

$$\bigwedge_{i=1}^{k} \bigwedge_{x \in R} \bigwedge_{\lambda \in \Lambda} (t^i > 0 \wedge \lambda^{i-1} = \lambda \Rightarrow (\mathit{ode}_\lambda(x)[\vec{X}^i/\vec{x}], \mathit{inv}_\lambda(x)[\vec{X}^i/\vec{x}], x^{i-1}, x^i, t^i)) \wedge \lambda^i = \lambda,$$

where $\mathit{ode}_\lambda(x)$ is the ODE pertaining to $x$ in $\lambda$ and $\mathit{inv}_\lambda(x)$ is the invariant pertaining to $x$ in $\lambda$. These constraints express that when residing in state $\lambda$ in step $i$ for a positive duration, the step-initial value $x^i$ and the flow-final value $x^i$ are connected by a flow of duration $t^i$ which satisfies $\mathit{ode}_\lambda(x)$ as well as $\mathit{inv}_\lambda(x)$.

*Reduction step 7.* The interplay between discrete states and transitions requires that $tr^i$ implies $\lambda^{i-1} = s(tr^i)$. This can be expressed by the $k \cdot |\mathit{Trans}|$ SSMT clauses in

$$\bigwedge_{i=1}^{k} \bigwedge_{tr \in \mathit{Trans}} \left(t^i = 0 \wedge tr^i = tr \Rightarrow \lambda^{i-1} = s(tr)\right).$$

Note that this condition is only enforced if the step duration is 0, which indicates a jump.

*Reduction step 8.* Likewise, *assignments* are dealt with by

$$\bigwedge_{i=1}^{k} \bigwedge_{tr \in \mathit{Trans}} \bigwedge_{\lambda' \in \Lambda} \left( \begin{array}{l} t^i = 0 \wedge tr^i = tr \wedge \lambda^i = \lambda' \Rightarrow \\ \mathit{asgn}_{tr,\lambda'}[x_1^i, \ldots, x_n^i / x_1', \ldots, x_n'][x_1^{i-1}, \ldots, x_n^{i-1} / x_1, \ldots, x_n] \end{array} \right)$$

which expresses that the flow-final values $\tilde{x}_j^i$ are connected to the step-final values $x_j^i$ by the assignment pertaining to transition $tr^i$.

*Reduction step 9.* We complete the encoding of the PHA transition dynamics by adding constraints describing the allowable initial states through the SSMT constraint system

$$\bigwedge_{\lambda\in\Lambda}\Big(\lambda^0=\lambda \;\Rightarrow\; init_\lambda[x_1^0,\ldots x_n^0/x_1,\ldots,x_n]\Big).$$

*Reduction step 10.* Finally, we complete the matrix by adding constraints describing the verification goal. In case we are interested in a probabilistic bounded reachability problem, this is that a state $(\lambda,\vec{x})$ with $\vec{x}\models Target_\lambda$ is eventually visited along the trace, i.e.

$$\bigvee_{i=0}^{k}\bigvee_{\lambda\in\Lambda}(\lambda^i=\lambda\wedge Target_\lambda[\vec{x}^i/\vec{x}]).$$

The conjunction of the above formulae yields the matrix of our SSMT formula encoding the PBMC problem. Satisfying valuations of the matrix thus obtained are in one-to-one correspondence to the runs of $A$ of length $k$ [31]. As in bounded model checking [16,23], satisfaction of temporal properties on all runs of depth $k$ can furthermore be checked by adding to the formula the $k$-fold unrolling [6] of a tableaux of the (negated) property, then checking the resulting formula for unsatisfiability. Using standard techniques from predicative semantics [37], the translation scheme can be extended to both shared variable and synchronous message-passing parallelism, thereby yielding formulae of size linear in the number of parallel components. This technique is used in the case studied presented in Section 6, where an exponentially more concise representation of shared-state concurrency is thus obtained.

*Phase 2: Encoding choices*: Let $\varphi$ be the matrix corresponding to the conjunction of the above formulae. As each non-deterministic choice corresponds to selecting a transition while each probabilistic choice amounts to selecting an actual target location, we generate the following SSMT formula:

*Reduction step 11.* An SSMT formula $\psi=\psi_1$ encoding the probabilistic and non-deterministic choices along the run is obtained by alternating the quantifiers consistently with the alternation of choices. To permit a homogeneous randomized quantification over all transitions, we select a finite set $O=\{o_1,\ldots,o_n\}$ of choice options for randomized choices, where $n$ is chosen minimal among the possible solutions of the following conditions, a probability distribution $p_O: O\xrightarrow{\text{total}}(0,1]$ over $O$, and a function $pd: Trans\times\Lambda\xrightarrow{\text{total}}\mathcal{P}(O)$ such that these together satisfy

$$\forall tr\in Trans,\lambda\in\Lambda:\sum_{pc\in pd(tr,\lambda)}p_O(pc)=p(tr)(\lambda)\ \text{ and}$$
$$\forall tr\in Trans,\lambda_1,\lambda_2\in\Lambda: pd(tr,\lambda_1)\cap pd(tr,\lambda_2)=\emptyset$$

Such a set $O$ and probability distribution $p_O$ exist always. The worst-case cardinality of $O$ is the number $|\{p(tr)(\lambda)\mid tr\in Trans,\lambda\in\Lambda\}|$ of different transition probabilities, but can be considerably smaller due to different probability constants being the sums of each other.

Now, we encode the non-deterministic choices by existential quantification over the transitions in *Trans* and the possible target states and we encode the probabilistic choices by randomized quantification over $O$. The latter quantifiers choose an auxiliary variable $pc_i$ in each step which in turn is mapped to the target location $\lambda^i$ by means of the mapping $pd$. Therefore, $\psi_i$ is defined recursively as follows: for $1\le i<k$,

$$\psi_i=\exists tr^i\in Trans:\exists x_1^i,\ldots,x_n^i:\mathfrak{R}_{p_O}pc_i\in O:\psi_{i+1}\ ,\ \text{and}$$
$$\psi_k=\varphi\wedge\bigwedge_{k=1}^{n}\bigwedge_{tr\in Trans}\bigwedge_{\lambda\in\Lambda}[(tr_i=tr\wedge\lambda^i=\lambda)\;\Rightarrow\;\bigvee_{o\in pd(tr,\lambda)}pc_i=o]$$

*Reduction step 12.* In order to solve the PBMC problem, it remains to choose the initial state maximizing the probability. This can be accomplished by existential quantification over the possible states, yielding the formula $PBMC^k_{A,Target}=\exists\lambda^0\in\Lambda:\exists x_1^0,\ldots,x_n^0\in\mathbb{R}:\psi$. Given the structural similarity between probabilistic bounded reachability and quantification in SSMT, this reduction is correct in the following sense:

[6] Involving dedicated termination rules for liveness properties [23].

**Proposition 1** (Correctness of reduction). *$Pr(PBMC^{k}_{A,Target}) \leq \theta$ iff A satisfies the PBMC problem w.r.t. threshold $\theta$, depth k, and target states Target.*

This proposition implies that by this reduction, we obtain a fully symbolic encoding of the PBMC problem and consequently, an SSMT solving algorithm permits analysis of the PBMC problem for dense-time probabilistic hybrid automata. In particular, an algorithm providing safe upper approximations of the satisfaction probability of SSMT formulae, as we will encounter in the next section, permits us to perform the following analysis tasks which are instantiations of the PBMC problem:

- Given a set *Bad* of hazardous states and a tolerable failure probability $\theta$, determine whether a step-bounded mission modelled as a PHA is sufficiently safe in the sense of its overall failure risk not exceeding the threshold $\theta$.
  This is a direct application of the above scheme for probabilistic bounded reachability, i.e. of the first version of the goals introduced in step 10.
- Given a linear-time temporal logic (LTL, [57]) formula $\phi$ characterizing safe or desired behavior and a step-bounded mission modelled as a PHA, determine whether it is safe in the sense of its overall probability of engaging into behaviors violating $\phi$ not exceeding the threshold $\theta$.
  This can be achieved by combining the above procedure with a bounded tableaux construction for LTL [23].
- Assessing reliability of a system subject to regular maintenance, where the number of system actions between maintenance is bounded by a constant *k*. Again, this is the same shape of proof, requiring to formulate the system dynamics as a PHA with the possible post-maintenance states as initial states.

## 5. Algorithm for SSMT problems

In this section, we present our algorithm for calculating the maximum probability of satisfaction of an SSMT formula. As indicated in Section 3, an exact and complete algorithm for solving SSMT formulae is impossible due to handling undecidable theories. We will provide an algorithm that, when queried "is the probability $Pr(\Phi)$ of satisfaction at most $\theta$?" for some SSMT formula $\Phi$ and some $\theta \in [0, 1]$, will provide reliable "yes" answers, yet may provide false negatives. I.e. whenever it claims "$Pr(\Phi) \leq \theta$" then this is reliable.

As a proof procedure, we generalize to SSMT over ODEs the extended Davis–Putnam–Logemann–Loveland (DPLL) algorithm [26,27] for SSAT described in [43]. The main idea of the DPLL-based approach to solving propositional SSAT problems $\Phi = Pre : \varphi$ is to enumerate all satisfying valuations of the quantifier-free SAT formula $\varphi$. Based on these solutions, the probability of satisfaction $Pr(\Phi)$ is computed in accordance with the semantic definition of SSAT (i.e. the special case of SSMT where all variables range over the Boolean domain, cf. Sect. 3.3). Clearly, such a naive enumeration approach establishes a decision procedure for SSAT but is far from being efficient since the number of valuations is exponential in the number of variables. Therefore, state-of-the-art implementations of SSAT algorithms employ a systematic search combined with various heuristics to gain performance. SSAT solvers manipulate partial valuations to the variables thus keeping track of the already visited search space. Furthermore, this gives the opportunity to immediately exclude partial valuations from the search that are meaningless for calculating the satisfaction probability, thus pruning away a potentially huge number of (total) valuations. Such pruning mechanisms are realized by, e.g. extending partial valuations by (logical) inferences (e.g. *unit propagation*, *pure variable elimination* [43]), skipping the alternative values of quantified variables (e.g. *threshold pruning* [43]), and storing and reusing the satisfaction probabilities of equivalent subformulae (*memoization* [46,48]). Another pruning technique is to shorten partial valuations upon detecting inconsistent or satisfying valuations which is referred to as *non-chronological backtracking* (e.g. [9,64]) or *solution-directed backjumping* [45], respectively. Intuitively, whenever a partial valuation was built up by freely *choosing* values for some quantified variables, in principle the alternative values need to be explored later on. These alternatives are stored as backtrack points. However, if it turns out that some of the alternative branches yield the same result as the current branch (i.e. an inconsistency or a solution), exploration of the alternative branch may be omitted for efficiency. Successively skipping *n* backtrack points prunes away $2^n$ partial valuations. The number of backtrack points to be skipped can be calculated from a so-called *reason* for an inconsistency or solution. For a comprehensive survey of SSAT algorithms and the listed heuristic enhancements the reader is referred to [47].

In contrast to SSAT the original definition of SSMT from [33,61] additionally allows non-quantified (interpreted as innermost existentially quantified) variables over *continuous domains* as well as *non-linear arithmetic constraints* like $x^3 < \sin(y)$. The domains of the randomized and existential variables –while not restricted to be Boolean– need to be finite. Though the abovementioned SSAT algorithm cannot be applied directly to SSMT problems, the basic idea however remains the same. The variables of the quantifier prefix are instantiated with values from left to right, thus building partial valuations. Once the whole quantifier prefix is processed, it remains to decide the satisfiability of a quantifier-free SMT formula. Such satisfiability problems are solved by an appropriate SMT solver. Since in this context the SMT formulae comprise non-linear arithmetic over the reals, we employ the SMT solver iSAT [32].

This approach to compute the satisfaction probability of SSMT problems is implemented in a SSMT solver called SiSAT [33, 61]. As mentioned above, enumerating all instantiations of the quantified variables cannot yield an efficient procedure for

practical applications. Therefore, the SiSAT tool implements all the algorithmic improvements known from state-of-the-art SSAT solvers which lead to performance gains of multiple orders of magnitude [cf. 61,62]. To show the practical significance of the SSMT-based symbolic model checking approach on realistic scenarios, we applied the SiSAT tool on a case study from the *networked automation system* (NAS) domain presented in [35]. The results for the NAS case study can be found in [62].

In this paper, we further enhance the expressive power of SSMT as indicated in Section 3. That is, the *additional* issues the enhanced SSMT procedure has to cope with are

1. to reason about ODE constraints and
2. to exhaustively explore quantifiers ranging over uncountable domains, as in existentially quantified continuous variables.

Addressing issue 1, we exploit overapproximation techniques based on safe interval calculations which permit safe reasoning by "enclosing" ODE solutions in interval-valued functions. These methods are described in Section 5.1. For solving constraint formulae containing an alternation of existential quantifiers ranging over the reals and of randomized quantifiers, the authors, to the best of their knowledge, know of no competing approaches. We address issue 2 by an algorithm which safely approximates the probability of satisfaction from above. The basic idea here is to exhaustively cover the domains of existential variables by small subintervals and reason about these using interval constraint propagation. This approach is safe in the sense that it will always deliver safe upper bounds of the probability of satisfaction (Section 5.2).

The presentation of the enhanced SSMT algorithm is organized in *three layers*. The lowermost layer is a combination of *theory solvers TS* for reasoning about a conjunctive system over the theory combination $T$ incorporating ODE constraints and arithmetic constraints. These theory solvers are interval-based, i.e. use interval calculations and interval constraint propagation [13] as safe, yet incomplete reasoning mechanisms. As the middle layer, an *SMT solver* for disjunctive systems over $T$ employs the theory solvers *TS*. Finally, the *SSMT solver* is an extension of the SMT layer to deal with existential and randomized quantification.

### 5.1. Reasoning about non-linear arithmetic and ODE constraints

We start our exposition with a description of the lowest layer of the solver, the *theory layer*, which deals with arithmetic and ODE constraints. Our approach to reasoning about such constraints is based on *interval arithmetic*. Therefore, instead of real- and integer-valued valuations, all base variables are interpreted over *interval valuations* $\rho : V_{base} \to \mathbb{I}_{\mathbb{R}} \cup \mathbb{I}_{\mathbb{Z}}$, where $V_{base}$ is the set of base variables of types integer and real as introduced before. $\mathbb{I}_{\mathbb{R}}$ and $\mathbb{I}_{\mathbb{Z}}$ are the sets of bounded intervals in $\mathbb{R}$ and in $\mathbb{Z}$, respectively – the Boolean domain can be represented by $\mathbb{B} = [0, 1] \subset \mathbb{Z}$. If both $\rho'$ and $\rho$ are interval valuations then $\rho'$ is called a *refinement* of $\rho$, denoted $\rho' \subseteq \rho$, iff $\rho'(v) \subseteq \rho(v)$ for each variable $v \in V_{base}$. Note that such interval valuations do not assign any value to the ODE variables $V_{ODE}$ as the existence of solution functions to the ODEs is handled locally in the theory layer for ODEs.

*Theory layer for non-linear arithmetic constraints*: As reasoning mechanisms for general non-linear constraint systems over the reals, we employ safe interval analysis (IA) for approximating real-valued satisfaction and augment it with interval constraint propagation (ICP) as a powerful deduction mechanism.

*Interval analysis* (e.g. [51]) enables to evaluate the *interval consistency* of a set $C_{arith}$ of non-linear arithmetic constraints involving functions like sin and exp. Interval consistency is a necessary yet not sufficient condition for real-valued satisfiability of $C_{arith}$. Thus, refutation by interval consistency is always correct. There are several definitions of interval consistency in the literature. They mainly differ in the strength of their consistency notions and in the computational effort to decide consistency. Our consistency concept is *hull consistency* [for details, cf. 13] up to a given accuracy, which is easy to decide and which gives good results in practice. Given an arithmetic constraint $a(\vec{x}) = b(\vec{x})$, where $a$ and $b$ are arithmetic terms over $\vec{x}$, and an interval valuation $\rho$ of its variables $\vec{x}$, interval arithmetic permits computing conservative interval approximations $\mathbb{I}_a(\vec{x})$ and $\mathbb{I}_b(\vec{x})$ of the ranges of $a(\vec{x})$ and $b(\vec{x})$ under $\rho$. If these intervals are not disjoint, i.e. if $\mathbb{I}_a \cap \mathbb{I}_b \neq \emptyset$, the constraint is called interval consistent. This notion can be lifted to conjunctive systems of constraints $C_{arith}$ requiring that each constraint $c \in C_{arith}$ is interval consistent under $\rho$.

*Interval constraint propagation* (e.g. [e.g. 12–14]) complements IA as a deduction mechanism pruning off non-solutions by narrowing the intervals while maintaining interval consistency. Given a constraint $c$ and an interval valuation $\rho$, ICP potentially computes a refinement $\rho' \subseteq \rho$ s.t. $\rho'$ contains all solutions of $c$ in $\rho$. Implementations of interval arithmetic support common functions like $+$, $-$, $\cdot$, as well as transcendental functions like sin [40,54]. For instance, if using floating-point data-types then intervals are always rounded outwards s.t. the results remain correct also under rounding errors. As an example, assume the arithmetic constraint $z = x + y$ and the interval valuation $\rho$ with $\rho(x) = [1, 4]$, $\rho(y) = [2, 3]$, and $\rho(z) = [0, 5]$ are given. Each solved form of the constraint, i.e. $x = z - y$, $y = z - x$, and $z = x + y$, allows contraction of the interval for the variable on the left-hand side. For $x = z - y$, we subtract the interval $[2, 3]$ for $y$ from the interval $[0, 5]$ for $z$, concluding that $x$ can only be in $[-3, 3]$. Intersecting this interval with the original interval $[1, 4]$, we know that $x$ can only be in $[1, 3]$. Proceeding in a similar way for $y = z - x$ does not change any interval, and finally, using $z = x + y$, we can conclude that $z$ can only be in $[3, 5]$.

We formalize these observations as follows. Let $M$ be a conjunctive system of constraints, $C_{arith} \subseteq M$ be the set of all arithmetic constraints in $M$, and $\rho$, $\rho'$ be interval valuations. The first two rules give the results of the interval consistency check while the third rule deduces a tighter interval valuation by ICP.

$$
\begin{aligned}
&(M, \rho) \longrightarrow_{IA} \texttt{cons} && \text{only if } C_{arith} \text{ is interval consistent under } \rho.\\
&(M, \rho) \longrightarrow_{IA} \texttt{incons} && \text{only if } C_{arith} \text{ is not interval consistent under } \rho.\\
&(M, \rho) \longrightarrow_{IA} (M, \rho') && \text{only if } \rho' \subseteq \rho \text{ and for each valuation } \sigma \in \rho, \sigma \notin \rho' :\\
& && \quad (M, \sigma) \longrightarrow_{IA} \texttt{incons}
\end{aligned}
$$

Note that the last line describes the ability of the theory layer to propagate tighter bounds by pruning off definite non-solutions.

*Theory layer for ODE constraints* In contrast to arithmetic constraints, which provide useful interval contractors already if considered in isolation (and applied alternatingly in a one-by-one fashion), ODEs require a more global view, as only definitionally closed sets of ODEs $\frac{dx}{dt} = f(x, y)$ and $\frac{dy}{dt} = g(x, y)$ are dynamically sufficiently constrained to provide a useful contractor. Therefore, we collect the ODE constraints from the conjunctive constraint system $M$ and apply the (costly) ODE processing to definitionally closed systems only. Given the set $M$, we take the subset $C_{ODE} \subseteq M$ of ODE constraints and call the elements $c_1, \ldots, c_m \in C_{ODE}$ (in no particular order) and the variables $x_1, \ldots, x_n \in V_{ODE}(\varphi)$, i.e. the ODE-defined variables. The task of the ODE theory layer is to decide whether, given a valuation $\rho$, there exists a refinement satisfying the ODE constraints, i.e. whether

$$\exists \sigma \in \rho : \exists y_1 \in \mathcal{C}^1_{fin}, \ldots, y_n \in \mathcal{C}^1_{fin} : eval(c_1, \sigma') \wedge \cdots \wedge eval(c_m, \sigma'), \tag{1}$$

where $\sigma' = \sigma \oplus [x_1 \mapsto y_1] \oplus \cdots \oplus [x_n \mapsto y_n]$ is a total valuation in the sense used in Section 3.2 while $\sigma$ itself does not contain any valuation for the variables from $V_{ODE}$. The ODE solver thus has to decide whether there exist solution functions $y_1, \ldots, y_n$ that satisfy the ODE constraints under one concrete valuation $\sigma'$ that is compatible to the current interval assignment $\rho$.

Actually *deciding* that question will be out of reach for most classes of ODEs but relaxing (1) yields an overapproximation similar to the one performed in the case of interval arithmetic for non-linear constraints. Assuming $\mathbb{I}_{\mathbb{R}}$ is the set of bounded intervals over $\mathbb{R}$ and $t_e \in \mathbb{R}_{\geq 0}$, we call $X : [0, t_e] \to \mathbb{I}_{\mathbb{R}}$ an *enclosure* of $x \in \mathcal{C}^1_{fin}$ iff $\forall t \in [0, t_e] : x(t) \in X(t)$ holds. Instead of using real-valued solutions functions for the ODE and checking the conditions on satisfied ODE constraints as laid down in Sect. 3.2.2, these conditions are checked for the enclosures $X_1, \ldots, X_n$. For example, having an ODE constraint $(ODE(x), I(x), u, v, \tau)$ and an interval valuation $\rho$ with $\rho(u) = [0, 1]$, a set of initial values, and $\rho(v) = [10.2, 12]$, a set of possible endpoints to be reached after $\rho(\tau) = [0, 5.2]$ time units, assume e.g. a constant enclosure $X(t) = [0, 5]$ for the solution trajectories of the ODE was calculated. Even though the exact solution trajectories emerging from $\rho(u)$ stay unknown, this enclosure is sufficient to decide that none of the possible endpoints $\rho(v)$ can be reached during the given time $\rho(\tau)$ because the intersection $X(\rho(\tau)) \cap \rho(v)$ is empty. We call a set of ODE constraints $C_{ODE}$ *inconsistent* with a valuation $\rho$ iff such enclosures $X_1, \ldots, X_n$ of the unknown exact solutions $x_1, \ldots, x_n$ can be calculated and there exists at least one constraint $c_i \in C_{ODE}$ that cannot be satisfied by any valuation $\sigma' \in (\sigma \oplus [x_1 \mapsto y_1 \in X_1] \oplus \cdots \oplus [x_n \mapsto y_n \in X_n])$ with $\sigma \in \rho$. Without knowing the exact solutions, the enclosures $X_1, \ldots, X_n$ are thus used to detect unsatisfiability of the constraints under $\rho$. Obviously, we never call a set $C_{ODE}$ inconsistent if (1) is actually satisfiable. This overapproximation is thus safe and its quality depends on the tightness of the enclosures $X_i$.

This task of deciding consistency and the task of propagating tighter bounds are therefore closely related to calculating safe enclosures of the solution sets of initial value problems. There are several techniques aiming at different classes of differential equations. The most "traditional" methods are based on extrapolating the state vector over time by interval-based evaluation of truncated Taylor series with an enclosure of the truncation error, safe handling of the rounding errors and some form of coordinate transformation to avoid the so-called wrapping effect. These methods date back to Moore [52] and were developed further by Lohner [44] and in their more recent more generalized incarnations are still considered competetive for linear ODEs [53]. In [29] we have presented a prototypical implementation of Taylor-series-based enclosures of ODE trajectories in our constraint solver iSAT. Taylor models [15] together with techniques called "shrink-wrapping" and "preconditioning" [49,50] to fight the wrapping effect allow tighter enclosures in the case of non-linear ODEs. Other methods include flowpipe enclosures using zonotopes [34], which explicitly allow to also handle bounded but otherwise unknown continuous inputs, and constraint-propagation-based methods like CLP(F) [39]. Along the lines of the integration approach we presented in [29], we will investigate the integration of some of these newer methods and use them as propagators for ODEs. In the following, we assume that a suitable enclosure method is available and we therefore concentrate on the correct use of it to solve the original question posed above.

In order to perform such an enclosure, *definitionally closed* systems of differential equations are required, i.e. for all variables $x$ occuring on the right-hand side of an ODE there must also be a constraint $c \in C_{ODE}$ with an $ODE(x)$ defining $x$. For variables $x \in V_{ODE}$ that do not occur in any of the constraints $c_1, \ldots, c_m$, we may assume arbitrary behavior, i.e. we do not need to care about such variables. In case a variable occurs on the right-hand side of an ODE but is not defined by

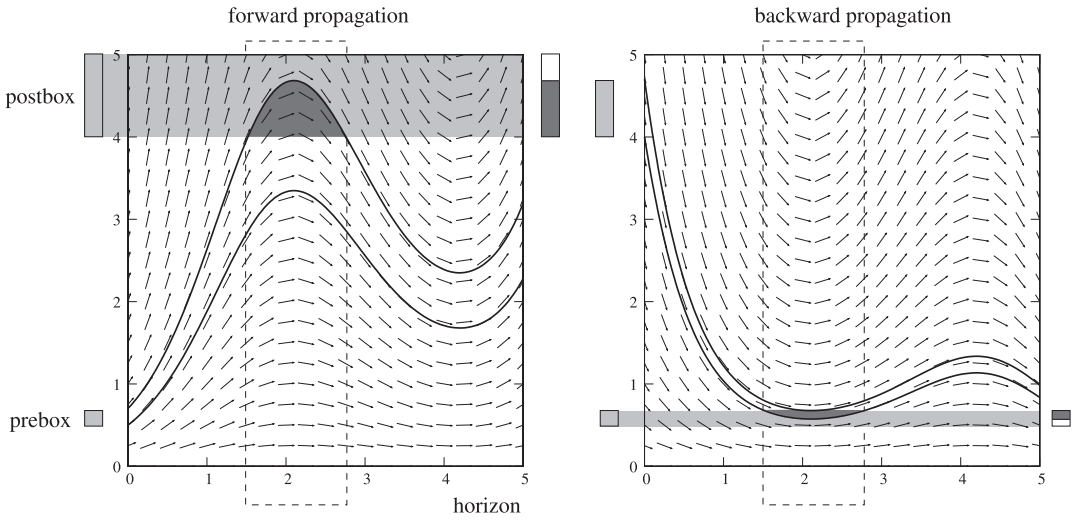


**Fig. 4**. Usage of ODE enclosures as pruning operators.

any constraint in $C_{ODE}$, we may either try to overapproximate all possible behaviors of that variable and then try to prove inconsistency or safely assume that the system is consistent, which is the much simpler option.

Neglecting the technical details, it is safe to assume that we can generate a number of definitionally closed sets of ODE constraints by grouping the constraints from $C_{ODE}$ together such that all variables occuring on the right hand side of the involved ODEs are themselves defined by ODE constraints. For the purpose of illustration, assume such a set is given by

$$S := \{(ODE(x_1), I(x_1), u_1, v_1, \tau_1), \ldots, (ODE(x_m), I(x_m), u_m, v_m, \tau_m)\}$$

which defines $m$ different variables $x_1, \ldots, x_m$ by ODE constraints. Ideally, this set is minimal in the sense that if any of the constraints were removed, $S$ would no longer be definitionally closed. Clearly this means that each of the variables defined by a constraint in $S$ occurs itself on the right-hand side one of at least one of the other constraints. For example the set $S = \{(ODE(x), I(x), a, b, t), (ODE(y), I(y), c, d, t)\}$ with $ODE(x) = \frac{dx}{dt} = x - y$ and $ODE(y) = \frac{dy}{dt} = y$ cannot be split up further into definitionally closed subsets as the right-hand side of $ODE(x)$ depends on $y$.

We call the box $\rho(u_1) \times \cdots \times \rho(u_m)$ the *prebox* and $\rho(v_1) \times \cdots \times \rho(v_m)$ the *postbox*. Taking the value $\Delta = \max(\bigcap(\rho(\tau_1), \ldots, \rho(\tau_m)))$ as a time *horizon*, we need to enclose all trajectories emerging from the prebox and satisfying all the differential equations in $S$ up to $\Delta$. If the intersection of this enclosure and the given postbox is empty on the entire interval $[\min(\bigcap(\rho(\tau_1), \ldots, \rho(\tau_m))), \max(\bigcap(\rho(\tau_1), \ldots, \rho(\tau_m)))]$, we know that no real-valued trajectories exist that connect any valuation from this pre- and postbox with the given length. We can therefore safely call $S$ and thereby also $C_{ODE}$ inconsistent with the given valuation $\rho$. Furthermore, if this intersection is not empty, we can remove all parts from the postbox that yield empty intersections with the computed enclosure, i.e. if the intersection of $\rho(v_1) \times \cdots \times \rho(v_m)$ with the enclosure is not empty, this intersection $\rho'(v_1) \times \cdots \times \rho'(v_m)$ may be used as a new valuation, as $\rho'$ still contains all possible solutions.

Note that we can use the same mechanism described above to enclose the set of forward-reachable states from $\rho(u_i)$ and prune $\rho(v_i)$, to also calculate the backwards-reachable set from $\rho(v_i)$ and prune off parts from $\rho(u_i)$. This is achieved by using the inverse ODE constraints and exchanging the roles of $u_i$ and $v_i$ in them. For each ODE constraint $\left(\frac{dx_i}{dt} = f_i(x_1, \ldots, x_n), I_i(x_i), u_i, v_i, \tau_i\right) \in C_{ODE}$, we can add an inverse constraint $\left(\frac{d\bar{x}_i}{dt} = -f_i(\bar{x}_1, \ldots, \bar{x}_n), I_i(\bar{x}_i), v_i, u_i, \tau_i\right)$, consider the variable $\bar{x}_1, \ldots, \bar{x}_n$ ODE-defined variables, and then perform the exact same steps as described above.

Fig. 4 visualizes this use of ODE enclosure mechanisms as propagators. Some of the trajectories emerging from the prebox reach the postbox during the temporal interval depicted by the dashed box. The upper part of the postbox, however, is not reached by any trajectory and can thus be safely pruned off. This tightened postbox serves as input to the backward propagation depicted in the right part of the figure. Using the inverse ODE, an enclosure over the same timespan for which the postbox was reachable during forward propagation – again indicated by the dashed box – now yields a tightened version of the prebox. Looking at the illustration of the forward propagation again, one can clearly see that exactly that part of the prebox was pruned off by backwards propagation from which no trajectory can actually reach the postbox.

Analogously to ICP, the above operations yield contractors that are able to narrow the intervals for variables $x$, $y$, $\tau$ in an ODE constraint $\left(\frac{dz_1}{dt}(t) = z_1 \cdot sin(z_2), \texttt{true}, x, y, \tau\right)$, as well as to potentially detect inconsistencies with the other constraints. These contractors yield a derivation relation $\longrightarrow_{ODE}$ yielding either narrowings or consistency information,

which satisfies the following properties:

$$
\begin{aligned}
&(M,\rho) \longrightarrow_{ODE} \texttt{cons} && \text{only if } \ C_{ODE} \text{ is not inconsistent under } \rho.\\
&(M,\rho) \longrightarrow_{ODE} \texttt{incons} && \text{only if } \ C_{ODE} \text{ is inconsistent under } \rho.\\
&(M,\rho) \longrightarrow_{ODE} (M,\rho') && \text{only if } \ \rho' \subseteq \rho \text{ and for each valuation } \sigma \in \rho, \sigma \notin \rho' :\\
& && \quad C_{ODE} \text{ is inconsistent under } \{\sigma\}.
\end{aligned}
$$

Note that this notion of an ODE constraint being "not inconsistent" with $\rho$ does not mean that $\rho$ actually contains a solution. If we call the constraint inconsistent under $\rho$, however, it definitely does not contain any solution.

*Combined theory layer*: Combining the theory solvers for non-linear arithmetic and ODE constraints yields the combined theory layer. Let $M$ be a set of non-linear arithmetic, denoted $C_{arith} \subseteq M$, and of ODE constraints, denoted $C_{ODE} \subseteq M$. Furthermore, let $\rho$, $\rho'$ be interval valuations. Then we merge both theory solvers into one, i.e. the interval consistency checks and the mechanisms of deducing tighter interval valuations.

$$
\begin{aligned}
&(M,\rho) \longrightarrow_{TS} \texttt{cons} && \text{iff } \ (M,\rho) \longrightarrow_{IA} \texttt{cons} \text{ and } (M,\rho) \longrightarrow_{ODE} \texttt{cons}\\
&(M,\rho) \longrightarrow_{TS} \texttt{incons} && \text{iff } \ (M,\rho) \longrightarrow_{IA} \texttt{incons} \text{ or } (M,\rho) \longrightarrow_{ODE} \texttt{incons}\\
&(M,\rho) \longrightarrow_{TS} (M,\rho') && \text{iff } \ (M,\rho) \longrightarrow_{IA} (M,\rho') \text{ or } (M,\rho) \longrightarrow_{ODE} (M,\rho')
\end{aligned}
$$

### 5.2. SSMT algorithm

In this section we present our algorithm for computing safe upper bounds on the probability of satisfaction of an SSMT formula. Our algorithm is an extension of the SiSAT algorithm described in [33,61] with existential quantification over continuous domains and with ODE constraints. The SiSAT approach generalizes the SSAT algorithm described in [43] based on the Davis–Putnam–Logemann–Loveland (DPLL) procedure [26,27].

*SMT layer*: The SMT layer is described by the following rules. More details on SMT can be found, e.g. in [7]. In contrast to the quantifier-free SMT setting, our SSMT formulae do not contain free variables of the base types real and integer. That is, the *branching* mechanism as well as the *backtracking* rule is not applied by the SMT solver but is executed by the SSMT layer which will be introduced later on. Thus, the SMT procedure covers only *theory propagation* and *theory consistency checking* combined with clause learning to exclude inconsistent valuations.

In the sequel, let $\varphi$ be an SMT formula in CNF over the theories of non-linear arithmetic and ordinary differential equations, $M$ be a conjunctive system of constraints from $\varphi$ which are asserted during the proof search, and $\rho$, $\rho'$ be interval valuations of the variables $V(\varphi)$. For conciseness in tracing reasons of deductions and thus learned conflicts, an interval valuation is represented as a list of interval bounds.

Rules R.1 and R.2 deduce new facts. Rule R.1 applies *unit propagation* if all constraints but one in a clause are inconsistent with $M$ under $\rho$, and adds the remaining constraint to be satisfied to $M$. *Theory propagation* is done by rule R.2 where new tighter interval valuations are deduced by ICP or by the ODE solver. Note that deriving empty intervals is forbidden. In case an empty interval could be deduced, the formula is inconsistent, which is covered by rule R.4

$$\frac{(c_1 \vee \ldots \vee c_m) \in \varphi, c_i \notin M, \forall j \neq i : (M \cdot \langle c_j \rangle, \rho) \longrightarrow_{TS} \texttt{incons}}{(\varphi, M, \rho) \longrightarrow_{SMT} (\varphi, M \cdot \langle c_i \rangle, \rho)} \tag{R.1}$$

$$\frac{(M,\rho) \longrightarrow_{TS} (M,\rho'), \forall x \in V(\varphi) : \rho'(x) \neq \emptyset}{(\varphi, M, \rho) \longrightarrow_{SMT} (\varphi, M, \rho')} \tag{R.2}$$

If a conflict occurs, i.e. the arithmetic constraints or the ODE constraints in $M$ are inconsistent under $\rho$, a more general interval valuation $\rho'$ with $\rho \subseteq \rho'$ as a *reason* for the conflict can be extracted. More formally, $\rho'$ is such a reason if there is a sequence of theory deductions from state $(M,\rho')$ to state $(M,\rho)$, i.e. $(M,\rho') \longrightarrow^*_{TS} (M,\rho)$. The interval valuation $\rho'$ can be symbolically encoded as a conjunctive system of interval bounds $b_1, \ldots, b_\ell$ where $b_i$ is of form $x \sim k$ with $x$ is a variable, $\sim \in \{<, \leq, \geq, >\}$, and $k \in \mathbb{R}$. We denote this symbolic encoding of $\rho'$ by $symb\_enc(\rho') = \{b_1, \ldots, b_\ell\}$. To prevent the SSMT layer from unnecessarily probing a refinement of $\rho'$ in future search, $\rho'$ is excluded by adding a so called *conflict clause* to the formula. Such a conflict clause can be easily constructed by the disjunction of the negated interval bounds $b_1, \ldots, b_\ell$, i.e. $(\neg b_1 \vee \ldots \vee \neg b_\ell)$. This new clause forbids any refinement of $\rho'$ in which a solution cannot exist by forcing that at least one of the bounds $b_1, \ldots, b_\ell$ may not hold. This is referred to as *conflict-driven clause learning* (rule R.3)

$$\frac{(M,\rho) \longrightarrow_{TS} \texttt{incons}, (M,\rho') \longrightarrow^*_{TS} (M,\rho), symb\_enc(\rho') = \{b_1, \ldots, b_\ell\}}{(\varphi, M, \rho) \longrightarrow_{SMT} (\varphi \wedge (\neg b_1 \vee \ldots \vee \neg b_\ell), M, \rho)} \tag{R.3}$$

Note that there are many different techniques for (efficient) generation of such an infeasible subsystem $b_1, \ldots, b_\ell$ of $\rho$. Very prominent in propositional SAT solving is the *first unique implication point technique* from [64]. Conflict-driven clause learning usually appears in combination with *non-chronological backtracking* (confer the introductory part of this section). However, in our setting the SMT layer does not know about possible backtrack points since all branching and backtracking steps are executed by the SSMT layer, in order to compute the satisfaction probabilities at backtrack points.

The last two rules of the SMT layer perform a consistency check. A proof state is inconsistent if the list of constraints $M$ which were asserted by rule R.1 are together inconsistent under the current interval valuation $\rho$

$$\frac{(M, \rho) \longrightarrow_{TS} \texttt{incons}}{(\varphi, M, \rho) \longrightarrow_{SMT} \texttt{incons}} \tag{R.4}$$

Note that rule R.3 also detects inconsistency but rule R.4 does not generate a reason for this conflict s.t. in concrete implementations of the algorithm conflict-driven clause learning can be applied according to heuristic measures or the ability of the theory solvers to provide reasons of conflict.

If each clause in $\varphi$ contains at least one constraint s.t. all these constraints together are consistent then the SMT formula is consistent

$$\frac{\exists M' \supseteq M\ \forall cl \in \varphi\ \exists c \in cl \text{ with } c \in M' : (M', \rho) \longrightarrow_{TS} \texttt{cons}}{(\varphi, M, \rho) \longrightarrow_{SMT} \texttt{cons}} \tag{R.5}$$

*SSMT layer*: The topmost SSMT layer is responsible for exhaustively branching the search space by choosing values and intervals for quantified variables, and for calculating the corresponding probabilities of satisfaction (cf. semantics of SSMT, Section 3.3). For the sake of clarity, we just present the *basic* proof rules of the SSMT layer to make clear the new contribution of handling existential quantification over continuous domains. That is, we do not mention the integration of pruning rules like *thresholding* or *solution-directed backjumping* which are already published and implemented in the SiSAT tool [33,61] and are indispensable for an efficient implementation. Such pruning rules cut off huge parts of the quantifier tree based on deductions at the SMT and SSMT layers, thus often avoiding the exponential blow-up of the search tree in quantifier depth.

Addressing issue 2 from the introduction to this section, we need to exhaustively cover the domain of existentially quantified continuous variables, i.e. compute the maximum over uncountably many alternatives. Algorithmically, this is infeasible in general. Thus, we aim at computing safe upper bounds $\theta \geq Pr(\Phi)$ of the actual probability of satisfaction.

We obtain safe upper bounds on existential quantification over bounded continuous intervals by partitioning the initial interval into subintervals of a given width $\varepsilon > 0$ and applying ICP to the resulting intervals. Formally, given a real-valued interval $\mathbb{I} = [l, u]$, *a partitioning of* $\mathbb{I}$ *w.r.t. some parameter* $\varepsilon$ is a finite set of disjoint intervals $P = \{[l, h_1], (h_1, h_2], \ldots, (h_k, u]\}$ s.t. the width of each interval in $P$ is at most $\varepsilon$. We instantiate the continuous variables by intervals of width $\varepsilon$ and consider interval consistency which overapproximates real-valued satisfaction s.t. if inconsistency is reported then no real-valued solution exists within the interval valuation. In case of proving interval consistency, a real-valued solution may exist. Consequently, we have a relaxed notion of satisfiability and obtain an upper bound on satisfaction probability.

In the sequel, we present the SSMT layer. Let $\Phi = Pre : \varphi$ be an SSMT formula and $\varepsilon$ some positive real-valued constant. We obtain a *partitioned w.r.t.* $\varepsilon$ quantifier prefix of *Pre* by replacing the real-valued interval domains $\mathrm{dom}(x)$ of all existentially quantified real-valued variables $x$ in *Pre* by partitionings $P$ of $\mathrm{dom}(x)$ w.r.t. $\varepsilon$, i.e. $\mathrm{dom}(x) := P$. A state of the SSMT solver is characterized by $(Pre, \varphi, M, \rho)$, where $\Phi = Pre : \varphi$ is an SSMT formula s.t. the domain of each existential real-valued variable in *Pre* is a set of intervals, $M$ is a list of asserted constraints, and $\rho$ is an interval valuation. Each deduction yields either a new proof state of the same structure, or a final state, i.e. a pair $(P, \varphi')$ of a probability and a new matrix. In the latter case the deduction chain terminates. If $(Pre, \varphi, M, \rho) \longrightarrow^*_{SSMT} (P, \varphi')$ then $P$ is an upper bound of the probability of satisfaction $Pr(Pre : \varphi)$. The new matrix $\varphi' \supseteq \varphi$ potentially contains learned conflict clauses, i.e. $\forall c \in \varphi' - \varphi : (\varphi \models c)$. Given an SSMT formula $\Phi = Pre : \varphi$ and some $\varepsilon > 0$ then the initial state is defined by $(Pre', \varphi, \emptyset, \rho)$ where $Pre'$ is a version of *Pre* partitioned w.r.t. $\varepsilon$, and $\rho$ is the smallest interval valuation which includes all initial variable domains.

The SSMT layer consists of the following rules. To compute the satisfaction probability according to the semantics of Section 3.3, we branch the proof search by *choosing* values or intervals (of width at most $\varepsilon$) from the domains of the quantified variables. Rules R.6 and R.7 apply if the leftmost quantifier is existential. While rule R.6 branches the search (domain size is at least two) and returns the maximum of both derived probabilities, rule R.7 computes the probability of the last alternative. Note that the domain $\mathrm{dom}(x)$ is either a set of values or of intervals. Both cases can be unified in one rule, respectively.

$$\frac{\begin{array}{c}|\mathrm{dom}(x)| \geq 2,\ v \in \mathrm{dom}(x),\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^*_{SSMT} (P_1, \varphi_1),\ consistent(\varphi_1, M, \rho),\\ (\exists x \in \mathrm{dom}(x) \setminus \{v\} \cdot Pre, \varphi_1, M, \rho) \longrightarrow^*_{SSMT} (P_2, \varphi_2)\end{array}}{(\exists x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (\max(P_1, P_2), \varphi_2)} \tag{R.6}$$

$$\frac{\begin{array}{c}|\mathrm{dom}(x)| = 1,\ v \in \mathrm{dom}(x),\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^{*}_{SSMT} (P, \varphi'),\ consistent(\varphi', M, \rho),\end{array}}{(\exists x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (P, \varphi')} \tag{R.7}$$

where $consistent(\varphi, M, \rho) := (\neg\exists cl \in \varphi : \forall c \in cl : (M \cdot \langle c\rangle, \rho) \longrightarrow_{TS} \texttt{incons})$ indicates whether the new matrix $\varphi$ is consistent with the list $M$ of asserted constraints and the interval valuation $\rho$. An inconsistency can occur, e.g. if the SMT layer adds a conflict clause to the matrix by rule R.3. In case of inconsistency, we do not need to probe the remaining alternatives of $x$, as these yield satisfaction probability 0. This will be handled by rules R.10 and R.11. The function $update(\rho, x = v)$ yields the interval valuation $\rho'$ s.t. for all variables $y \neq x : \rho'(y) = \rho(y)$, and $\rho'(x) = v$ if $v$ is an interval and $\rho'(x) = [v, v]$ if $v$ is a value.

Analogously, the branchings rules for the randomized quantifier can be formalized. Here, the domains always consist of finitely many discrete values, and the resulting probability is the weighted sum

$$\frac{\begin{array}{c}|\mathrm{dom}(x)| \geq 2,\ v \in \mathrm{dom}(x),\ (v \to p_v) \in d,\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^{*}_{SSMT} (P_1, \varphi_1),\ consistent(\varphi_1, M, \rho),\\ (\mathfrak{R}_d x \in \mathrm{dom}(x) \setminus \{v\} \cdot Pre, \varphi_1, M, \rho) \longrightarrow^{*}_{SSMT} (P_2, \varphi_2)\end{array}}{(\mathfrak{R}_d x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (p_v \cdot P_1 + P_2, \varphi_2)} \tag{R.8}$$

$$\frac{\begin{array}{c}|\mathrm{dom}(x)| = 1,\ v \in \mathrm{dom}(x),\ (v \to p_v) \in d,\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^{*}_{SSMT} (P, \varphi'),\ consistent(\varphi', M, \rho)\end{array}}{(\mathfrak{R}_d x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (p_v \cdot P, \varphi')} \tag{R.9}$$

It may happen that after an SSMT call for branch $x = v$, all remaining branches for variable $x$ lead to inconsistent interval valuations and thus to satisfaction probability 0. In such a case, we may immediately skip probing all alternative values of $x$ while returning the (weighted) satisfaction probability for branch $x = v$. In order to find out whether such a situation is present, we search for inconsistent conflict clauses in the new matrix $\varphi'$ returned by the SSMT call for branch $x = v$. Such conflict clauses could be added to the old matrix $\varphi$ by rule R.3 during exploration of branch $x = v$. A clause $cl \in \varphi'$ is inconsistent with the current solver state if each constraint $c \in cl$ is inconsistent with $M$ and $\rho$, i.e. $(M \cdot \langle c\rangle, \rho) \longrightarrow_{TS} \texttt{incons}$. If a clause in $\varphi'$ is inconsistent then the conjunction of clauses $\varphi'$ is inconsistent. Inconsistency of a clause in $\varphi'$ is formalized by $inconsistent(\varphi', M, \rho) := \neg consistent(\varphi', M, \rho)$. Observe that under any extension $M'$ of $M$ and any refinement $\rho'$ of $\rho$ the matrix $\varphi'$ remains inconsistent, i.e. $inconsistent(\varphi', M', \rho')$ is true. This implies that the satisfaction probabilities of all remaining branches for $x$ are 0. Rules R.10 and R.11 save unnecessary visits of these branches.

$$\frac{\begin{array}{c}v \in \mathrm{dom}(x),\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^{*}_{SSMT} (P, \varphi'),\ inconsistent(\varphi', M, \rho),\end{array}}{(\exists x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (P, \varphi')} \tag{R.10}$$

$$\frac{\begin{array}{c}v \in \mathrm{dom}(x),\ (v \to p_v) \in d,\\ (Pre, \varphi, M, update(\rho, x = v)) \longrightarrow^{*}_{SSMT} (P, \varphi'),\ inconsistent(\varphi', M, \rho)\end{array}}{(\mathfrak{R}_d x \in \mathrm{dom}(x) \cdot Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (p_v \cdot P, \varphi')} \tag{R.11}$$

Note that chained executions of that rules, which occur if the returned matrix $\varphi'$ is also inconsistent on some previous levels, correspond to *non-chronological backtracking*.

All of the aforementioned SSMT rules are designed to deal with existential and randomized quantification. The following rules embed the SMT layer into the SSMT algorithm. If the SMT solver can propagate new facts by rule R.1 (*unit propagation*), rule R.2 (*theory propagation*), or rule R.3 (*clause learning*), we lift these deductions to the SSMT layer

$$\frac{\begin{array}{c}(\varphi, M, \rho) \longrightarrow_{SMT} (\varphi', M', \rho'),\\ Pre' = reduce(Pre, \rho')\end{array}}{(Pre, \varphi, M, \rho) \longrightarrow_{SSMT} (Pre', \varphi', M', \rho')} \tag{R.12}$$

The function $reduce(Pre, \rho')$ returns a modified prefix $Pre'$ s.t. all values and (sub)intervals are removed from the variable domains which are inconsistent with the new interval valuation $\rho'$.

Whenever the domain of some variable becomes empty by either quantifier-domain reduction in rule R.12 or when the current SSMT state is inconsistent within the SMT layer, we know that $\varphi$ is unsatisfiable over any refinement of the current valuation. Thus, we immediately return probability 0

$$\frac{\exists(Qx \in \mathrm{dom}(x)) \in \mathit{Pre}\ \text{s.t.}\ \mathrm{dom}(x) = \emptyset \quad \text{or} \quad (\varphi, M, \rho) \longrightarrow_{\mathit{SMT}} \texttt{incons}}{(\mathit{Pre}, \varphi, M, \rho) \longrightarrow_{\mathit{SSMT}} (0, \varphi)} \tag{R.13}$$

If all quantified variables are instantiated, i.e. the prefix is empty, and the SMT rule R.5 reports consistency, existence of a satisfying real-valued assignment cannot be refuted by the SMT layer. Thus, we return the upper bound 1 on probability

$$\frac{(\varphi, M, \rho) \longrightarrow_{\mathit{SMT}} \texttt{cons},}{(\varepsilon, \varphi, M, \rho) \longrightarrow_{\mathit{SSMT}} (1, \varphi)} \tag{R.14}$$

### 5.3. Soundness and termination of the SSMT algorithm

The algorithm to address SSMT problems was presented in a rule-based form, leaving implementation freedom w.r.t., e.g. different branching heuristics in the SSMT layer, different conflict clause learning schemes in SMT, or the choice of different theory solvers. We prove soundness and termination of the SSMT algorithm relative to soundness and termination of the SMT solver and the underlying theory solvers. For proofs of such properties for SMT approaches with decidable background theories like linear arithmetic, and for the undecidable case of non-linear arithmetic involving transcendental functions, the interested reader is referred to [55] and [32], respectively.

**Proposition 2** (Soundness). *Assume that each SMT rule $(\varphi, M, \rho) \longrightarrow_{\mathit{SMT}} S$ is sound in the sense that*

- *if $S = \texttt{incons}$ then there is no assignment in $\rho$ which satisfies the constraint system $M$, and*
- *if $S = (\varphi', M', \rho')$ then $\varphi \Rightarrow \varphi', \varphi \wedge M \Rightarrow M'$, and there is no assignment in $\rho \setminus \rho'$ which satisfies the constraint system $M$.*

*Let $\Phi = \mathit{Pre} : \varphi$ be an SSMT formula, $\varepsilon > 0$ be some real-valued constant, $\mathit{Pre}'$ be a partitioned version of Pre w.r.t. $\varepsilon$, and $\rho$ be the smallest interval valuation which includes all initial domains of the variables $V(\varphi)$. Then, if $(\mathit{Pre}', \varphi, \emptyset, \rho) \longrightarrow^{*}_{\mathit{SSMT}} (P, \varphi')$ then $Pr(\Phi) \leq P$.*

**Proof.** First observe that rule R.12 is based on semantic inferences, which are by assumption sound and constrain the search space such that non-solutions of $\varphi$ are removed. By the definition of SSMT, non-solutions yield satisfaction probability 0, which is taken into account by the backtrack rules R.10 and R.11, and by the inconsistency rule R.13.

In the base cases, i.e. if inconsistency is detected or the quantifier prefix is empty, safe upper bounds for the probabilities are computed by rules R.13 and R.14. (Note that non-strict upper bounds may be introduced by rule R.14 since we cannot decide satisfaction here in general.) Now assume the probabilities in the premises of rules R.6–R.11 are safe upper bounds. Then each of these rules computes safe upper probability bounds according to the semantics definition. This gives us the inductive argument completing the proof. □

For termination of the SSMT algorithm, we must limit the number of consecutive SMT propagation steps in rule R.12, as deduction mechanisms like interval constraint propagation are not terminating in general. A usual approach in practice enforcing such termination is to stop propagation if the progress in the interval narrowing becomes negligible. Relative to such enforced termination at the SMT layer, termination of the SSMT layer is easy to show: each SSMT rule except rule R.12 leads directly to a final state. It remains to be shown that checking its premise is terminating. The latter holds by induction since SSMT-calls in a premise are done on smaller problems (either the leftmost element from the finite quantifier prefix or an element from the finite domain of the leftmost quantified variable is removed). The base cases are given by rules R.13 and R.14.

### 5.4. Example of the SSMT algorithm

Consider the SSMT formula $\Phi =$

$$\exists x \in [-0.5, 1.5]\ \Re_{[1 \to 0.3, 2 \to 0.7]} y \in \{1, 2\} :$$

$$(x^2 \leq 25.3) \wedge (x > 0 \vee y = 1) \wedge (x \leq 0 \vee y = 2)$$

from Fig. 3 which is slightly modified by a smaller interval for $x$ for illustration reasons. Note that this does not affect the probability of satisfaction. Given the parameter $\varepsilon = 0.5$, a partitioned domain of $x$ w.r.t. $\varepsilon$ is the set $D = \{[-0.5, 0], (0, 0.5], (0.5, 1], (1, 1.5]\}$ of intervals, all of width 0.5. Then, the initial proof state is

$$(\exists x \in D\, \mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \emptyset, \rho)$$

where $\varphi = (x^2 \leq 25.3) \wedge (x > 0 \vee y = 1) \wedge (x \leq 0 \vee y = 2)$, and $\rho$ is given by $\rho(x) = [-0.5, 1.5]$, $\rho(y) = [1, 2]$. By rule R.12, we can execute SMT unit propagation (R.1), i.e.

$$(\varphi, \emptyset, \rho) \longrightarrow_{SMT} (\varphi, \langle x^2 \leq 25.3\rangle, \rho)$$

since the first clause is initially unit. This may be followed by SMT theory propagation (R.2). However, from $x^2 \leq 25.3$, ICP cannot conclude tighter intervals for $x$. Note that for a greater interval of $x$, like $[-9.8, 10.1]$ as in Fig. 3, ICP would contract it to $[-5.03, 5.03]$. By rule R.12, we enter the new SSMT state

$$(\exists x \in D\, \mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \langle x^2 \leq 25.3\rangle, \rho)$$

Now just rule R.6 or R.10 are applicable. To find out which rule matches, we choose an interval for $x$, say $[-0.5, 0]$. The new state is

$$(\mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \langle x^2 \leq 25.3\rangle, \rho_1)$$

where $\rho_1(x) = [-0.5, 0]$ and $\rho_1(y) = \rho(y)$. Then we can perform unit propagation (R.1) for the second clause $(x > 0 \vee y = 1)$, since the theory constraint $x > 0$ is inconsistent under $\rho_1$.

$$(\varphi, \langle x^2 \leq 25.3\rangle, \rho_1) \longrightarrow_{SMT} (\varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_1)$$

Theory propagation (R.2) narrows the interval of $y$.

$$(\varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_1) \longrightarrow_{SMT} (\varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_2),$$

where $\rho_2(y) = [1, 1]$ and $\rho_2(x) = \rho_1(x)$. Hence, two executions of rule R.12 lead to state

$$(\mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1\}, \varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_2)$$

Note that the domain of $y$ is also reduced in the prefix. Then there is just one possible value to choose for the randomized variable $y$. Now take rule R.9, where in its premise rule R.14 is applicable, since no inconsistency can be detected.

$$(\emptyset, \varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_2) \longrightarrow_{SSMT} (1, \varphi)$$

By rule R.9, we have probability $0.3 = 0.3 \cdot 1$.

$$(\mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1\}, \varphi, \langle x^2 \leq 25.3, y = 1\rangle, \rho_2) \longrightarrow_{SSMT} (0.3, \varphi)$$

Here, we know that we have to apply rule R.6 for state

$$(\exists x \in D\, \mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \langle x^2 \leq 25.3\rangle, \rho)$$

if we select interval $[-0.5, 0]$ for $x$. To abridge the example, we assume that in the premise of rule R.6, the following deduction sequence holds.

$$(\exists x \in D \setminus \{[-0.5, 0]\}\, \mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \langle x^2 \leq 25.3\rangle, \rho) \longrightarrow^*_{SSMT} (0.7, \varphi)$$

Then we can compute the maximum of the probabilities calculated in the premise, i.e. $\max(0.3, 0.7) = 0.7$.

$$(\exists x \in D\, \mathfrak{R}_{[1\to 0.3, 2\to 0.7]} y \in \{1, 2\}, \varphi, \langle x^2 \leq 25.3\rangle, \rho) \longrightarrow_{SSMT} (0.7, \varphi)$$

Hence, an upper bound probability for the SSMT formula $\Phi$ is 0.7, which here also is the actual probability of satisfaction.

## 6. Experimental evaluation of the algorithm

In order to evaluate the algorithm proposed in this paper, we have integrated our current work on ODE enclosure methods, which extends the approach presented in [29], into the SiSAT tool as described in [61,62]. This prototypical implementation still lacks some of the features presented in this paper; chief among them, it does not support existential quantification over continuous domains and its ODE propagation currently only works in forward direction. In constrast to the approach to embedding ODE enclosures as propagators into the iSAT algorithm described in [29], we use Nedialkov's VNODE-LP [53] as the underlying enclosure mechanism. The reimplementation of our ODE enclosure layer necessitated by this change is the reason why the tool still lacks support for backwards propagation, intersection with flow invariants, and several technical optimizations but, on the other hand, allows learning of deduced enclosures in new clauses – similar to the conflict-driven clause learning rule R.3 described in Section 5.2.

### 6.1. Case study

In the remainder of this section, we present the quantitative analysis of a networked control system, starting with an abstract system description and an automaton model of that system, succeeded by the encoding as an SSMT formula and concluded with the experimental results obtained when making use of our prototypical implementation. Though the scenario of this case study is of academic nature, it is representative of a number of real-world industrial applications comprising the interplay of controlled continuous processes with feedback control mediated by a communication network exhibiting communication latencies. Among the frequently used communication protocols in such environments is the class of carrier sense multiple access protocols with collision detection (CSMA/CD) or with randomized collision avoidance (CSMA/CA), which are popular because of their relatively low average-case latency and the widespread availability of corresponding network components. The price to be paid is a broad variation in actual communication latencies stemming from retries after collision detection or random retreat in collision avoidance. The resulting large jitter in the end-to-end latency of the feedback path poses a major problem in networked control systems. A classical countermeasure is a pessimistic and thus costly dimensioning of the communication network, which results from both the aim of generally keeping the jitter low and the limited ability of current analysis methods to analyze such systems in their entirety. In the following, we demonstrate that the methods proposed in this paper are in principle suitable for such an analysis. In particular, manipulating a holistic model of the networked control loop, they cover not only the effects of jitter on the controlled system, but also the otherwise often neglected dependency of communication frequency and thus collision probability and jitter on the state of the controlled system.

The case study of this paper is illustrated in Fig. 5. Two processes are to be regulated by a networked control unit. These processes of potentially physical, biological, or chemical nature are continuously evolving while the actuators may occasionally switch some of their controlled inputs, causing a change of the processes' continuous behaviors. Periodically, the observable states of the processes are measured by the sensors and then transmitted to the controller via the communication bus. The controller checks whether the current state of a process requires some modification of its inputs to change the process' behavior. If so, the controller sends an adequate data package over the bus to the corresponding actuator. In the present scenario we assume the bus to be a shared medium subject to collisions, i.e. only one of the sensors or the controller may use the bus at a particular time. Obviously, this restriction leads to situations where a device is ready to send but cannot actually start sending due to bus occupancy, and also to situations where messages get lost due to simultaneous sending of more than one device. In both cases the protocol tries to resolve the undesired situation by random retreat, i.e. through assigning a random delay before resend to each of the conflicting senders. Given such a randomized protocol

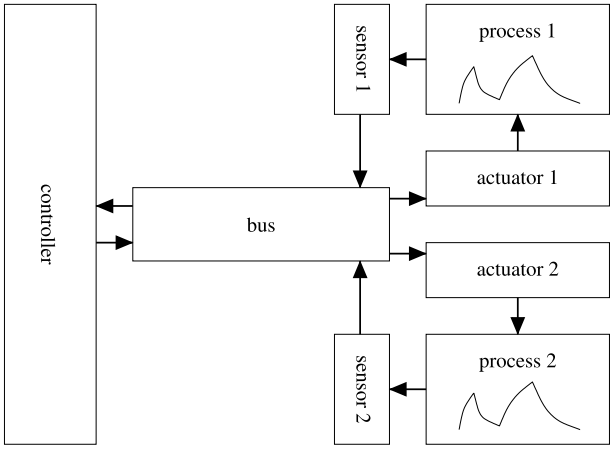


**Fig. 5**. A networked control system.

embedded in the described application, a quantitative analysis of the overall system requires a precise exploration of all the possible interleavings between the continuously evolving processes, the timing delays of the discrete components, and the probabilistic choices in resolving potential communication conflicts.

### 6.2. Parallel components model of the case study

An implementation of the abstract case study motivated above is shown in Figs. 6 and 7. We have modelled the system by a set of parallel automata, each reflecting particular aspects of the system dynamics. Note that flattening out this concurrent system by a product construction would yield a PHA of $2^6 \cdot 3^2 \cdot 5^3 \cdot 7 = 504.000$ graphical locations and, due to two further Boolean variables signalling successful sends, slightly more than 2 million discrete locations overall. In Fig. 6, the plant model with its continuous evolution is given by two hybrid automata representing the two processes p1 and p2. If left uncontrolled, each of the processes converges exponentially to either +10 or -10, depending on the mode it is in. The task of the controller will be to keep the system inside the corridor $[-8.8, 8.8]$ by switching between the modes early enough. Therefore sensors, depicted by the automata s1 and s2, regularly try to send current measurements of the continuous variables $x_1$ and $x_2$ representing the continuous state of the plant over the bus to the controller. The controller (cf. Fig. 7) comprises a receiver part for each of the sensors (cr1 and cr2), an arbiter (ca) to queue pending messages to be sent to the actuators, and a sender (cs) whose task is to transmit the actual messages to the actuators. In order to compensate for the anticipated network delay, the controller already decides to send a switch message whenever a received sensor value falls outside the corridor of $[-8.0, 8.0]$. Successfully receiving such a switch message from the controller, the actuators (a1 and a2) perform the switch – modelled by sending an event p1_toggle or p2_toggle to the respective process p1 or p2.

As the bus protocol is one of the central aspects of the model, a more detailed explanation is of interest. As mentioned above, only one device can successfully transmit a message over the bus at a particular time. If more than one sender is using the bus, this can be detected afterwards – in practice by, e.g. bus snooping or through lack of an acknowledge message from the receiver addressed. This behavior is modelled by the bus_error automaton, which switches to its error state as soon as there is more than one sender and leaves it only after the last of these senders has ceased sending. Thus, also the last sender can detect that its message has not been transmitted successfully.

According to the implemented protocol, a device has to sense the bus prior to actually starting to send in order to avoid unnecessary conflicts. If the bus is not free when sensing or if an error occurred during transmission, a random retreat scheme is employed by setting a timer to a random value and waiting for the corresponding duration. These random delays can be seen in the automata on the probabilistic transition from states sense to wait and send to wait respectively. For sensor s1, these are, e.g. 4 time units (t.u.) chosen with probability 0.25, 7 t.u. (probability 0.75), and 13 t.u. (probability 0.25). We chose lower delays for the controller because its messages have both a higher urgency and a lower frequency: it only sends a message if a toggle is really required. The sensor messages on the other hand will often contain measurements that will not cause any further reaction, as they will mostly frequently indicate that the current control mode can be held.

### 6.3. Encoding of the system

Each of the automata described above has been manually encoded into SSMT format. Thereby, we have followed the scheme described in Section 4, yet have deviated from strictly applying that scheme where simplifications in the encoding were obvious. For example, all transitions in the components of the system have deterministic guard conditions such that there no need for existential quantification upon transition selection. As introduced in Section 4, each instance of the variables in the predicative encoding describes the system's state at a particular point of time. Therefore, parallelism demands that each automaton can be interrupted and perform a transition at any moment, thereby keeping the valuation of its variables constant (often called *stutter jumps*). The formula thus explicitly encodes such self-loops that are invisible in the automata depicted in the figures. The different forms of communication between the automata (using events, shared variables, or state observation) are all mapped to using the valuation of the corresponding variables in those parts of the formula encoding transition guards or actions. To compute the duration between two successive snapshots in a *scheduled event* fashion, we introduce a step variable for each of the components that is set to zero whenever the automaton switches between modes (which takes no time) or set to the anticipated flow duration when the automaton is known to reside in a mode (e.g. when a sensor is waiting). In the latter case, the *next* variables (encoding the temporal position of the next event) are used to determine these durations. The duration of the global step is thus the minimum of the local steps. For more details, we refer the reader to the input files that contain the complete encoding. [7]

To evaluate our proposed integration of safe ODE enclosures into the SSMT algorithm, we compare three different encodings of the continuous behavior. First, we encode the differential equation directly in order to have it handled by the combined ODE-SSMT solver. As for this example, the closed-form solutions are easily computable, in a second encoding the ODE constraints are replaced by the explicit solution functions, thus forming an SSMT model without any ODE constraints. As such solution functions are in general not easily obtainable, our third model comprises a safe overapproximation of the continuous behavior. In order to achieve this, we calculate the first Taylor terms of the ODE's exact solution and use the

[7] To be found at `http://www.avacs.org/Benchmarks/Open/sisat_ode_benchmark.tar.gz`.

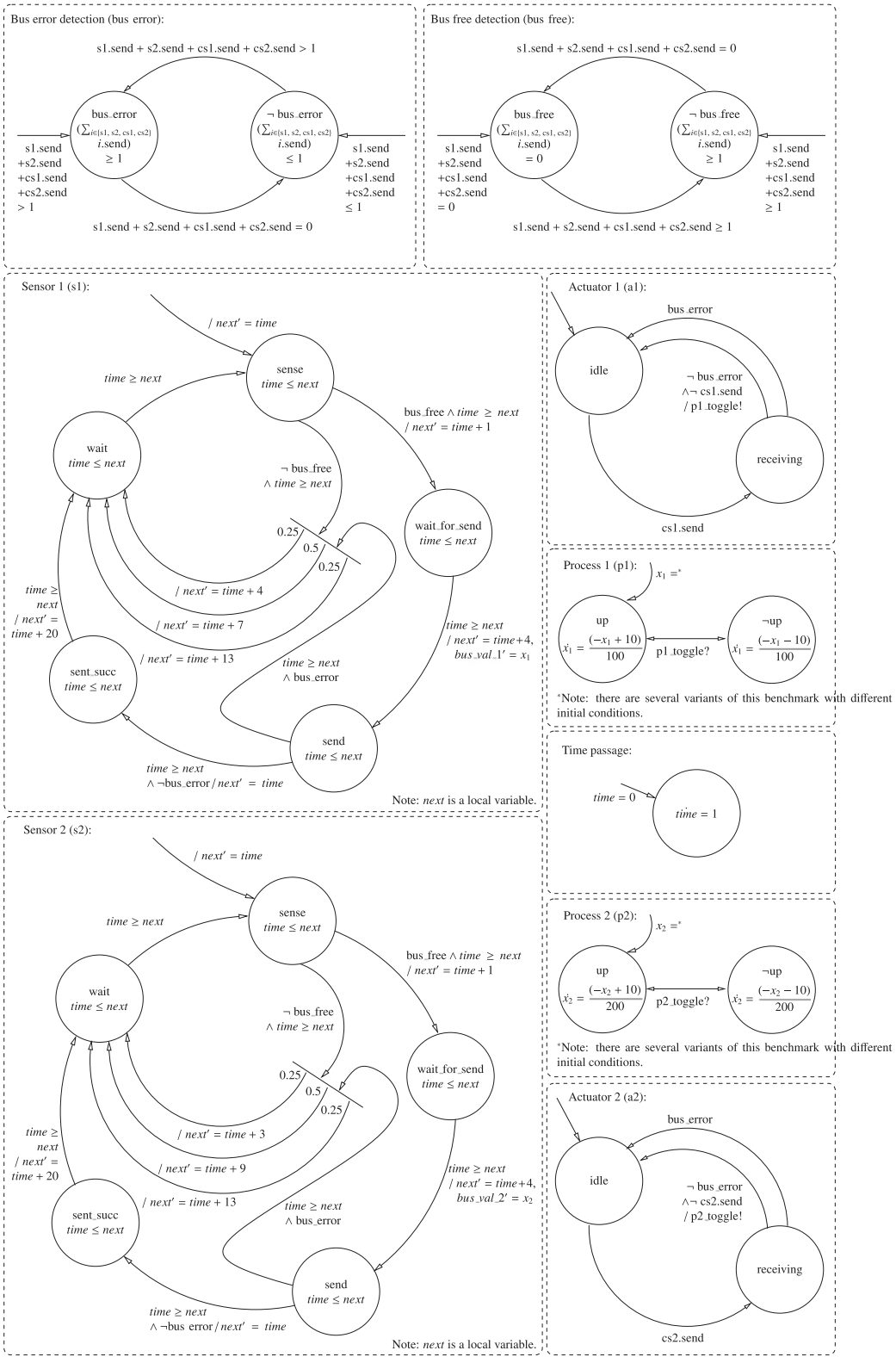


**Fig. 6**. Parallel automata of the case study: bus, sensors, actuators, and continuous processes.

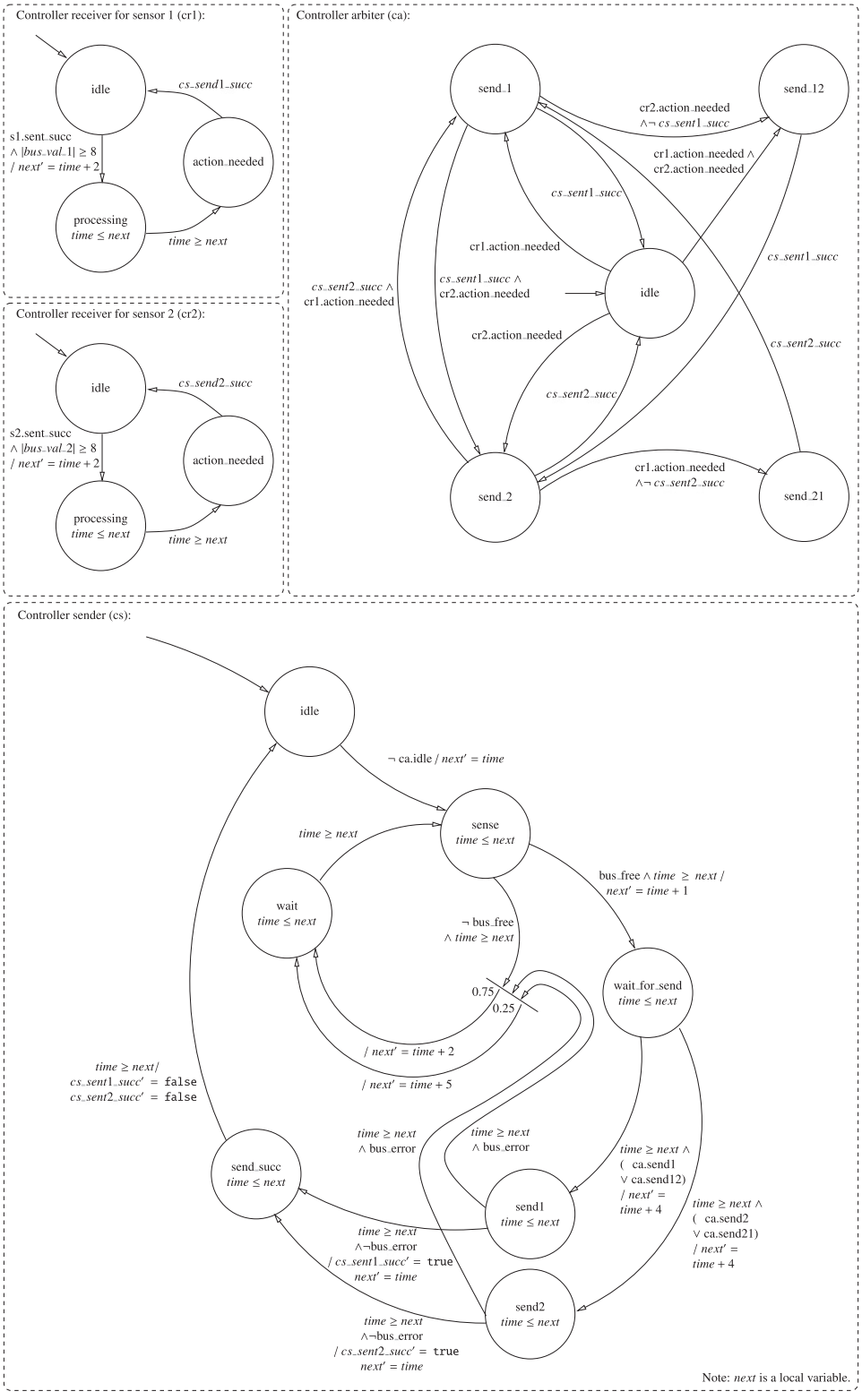


**Fig. 7**. Parallel automata of the case study: parts of the controller.

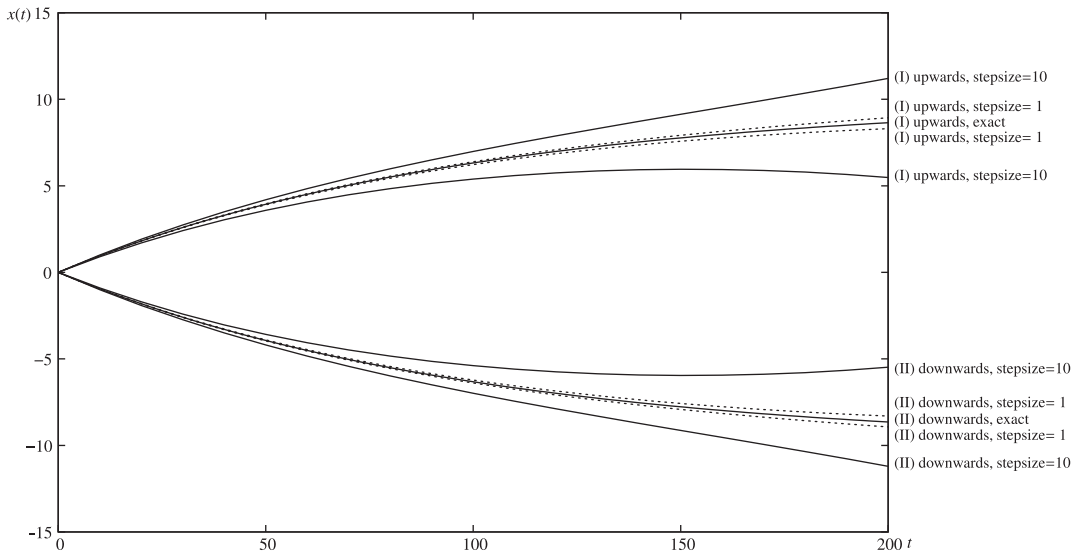


**Fig. 8**. Overapproximation compared to exact solution.

knowledge that any trajectory starting within $[-10, 10]$ will never leave that interval to bound the error term. The resulting formula thus performs an Euler step and compensates for the truncation error using the bounded second order Taylor term. For the ODEs

$$\frac{d\,x}{d\,t} = \frac{-x(t) + 10}{r}, \text{ with } r \in \{100, 200\} \quad \text{(I)}$$

$$\frac{d\,x}{d\,t} = \frac{-x(t) - 10}{r}, \text{ with } r \in \{100, 200\} \quad \text{(II)}$$

the solution functions are given by

$$x(t_0 + h) = \quad 10 + e^{-h/r} \cdot (-10 + x(t_0)) \quad \text{for (I)}$$

$$x(t_0 + h) = -10 + e^{-h/r} \cdot (\quad 10 + x(t_0)) \quad \text{for (II)}$$

and the overapproximations by

$$x(t_0 + h) \in x(t_0) + h \cdot \frac{-x(t_0) + 10}{r} + \frac{h^2}{2} \cdot \frac{-[0, 20]}{r^2} \quad \text{for (I)}$$

$$x(t_0 + h) \in x(t_0) + h \cdot \frac{-x(t_0) - 10}{r} + \frac{h^2}{2} \cdot \frac{-[-20, 0]}{r^2} \quad \text{for (II)}$$

using $[-10, 10]$ as a safe overapproximation for $x(t)$ in the remainder term. Fig. 8 illustrates the enclosure characteristics of this overapproximation and its conservativeness for different stepsizes $h$ and $r = 100$.

In order to compare this overapproximation with VNODE-LP [53], which we use internally to enclose the ODE trajectories, we called the VNODE-LP tool with the given ODE (I) for upwards dynamics and $r = 100$. Starting for a point value $x(0) = 0$, the tool was able to take one step of length 200 to reach the final point directly, calculating a safe enclosure for that point $x(200) = 8.6466471676338[674, 799]$ in a small fraction of a second. Computing tight interval enclosures for interval-valued initial conditions, e.g. given $x(0) \in [0, 3]$, yields the enclosure $x(200) \in [8.6466471676338674, 9.0526530173437188]$, which is exact up to at least 14 digits – again using only one direct step.

While these three different encodings can be used to reveal the impact of using an ODE solver inside the SiSAT algorithm, an additional impact factor may be the level of random and existential quantification as well as newly introduced non-determinism. We therefore define four scenarios featuring different characteristics regarding these criteria:

1. By replacing the random choices in the model by pre-defined constant values and setting the initial values of $x_1$ and $x_2$ to 0, the model becomes completely deterministic. We use this model to validate the behavior of the system.
2. Again fixing the initial values of $x_1$ and $x_2$ to 0 but leaving the randomized quantification intact, we get the smallest model that reflects the described behavior, i.e. faithfully represents collisions on the bus and the consequences of random retreat.

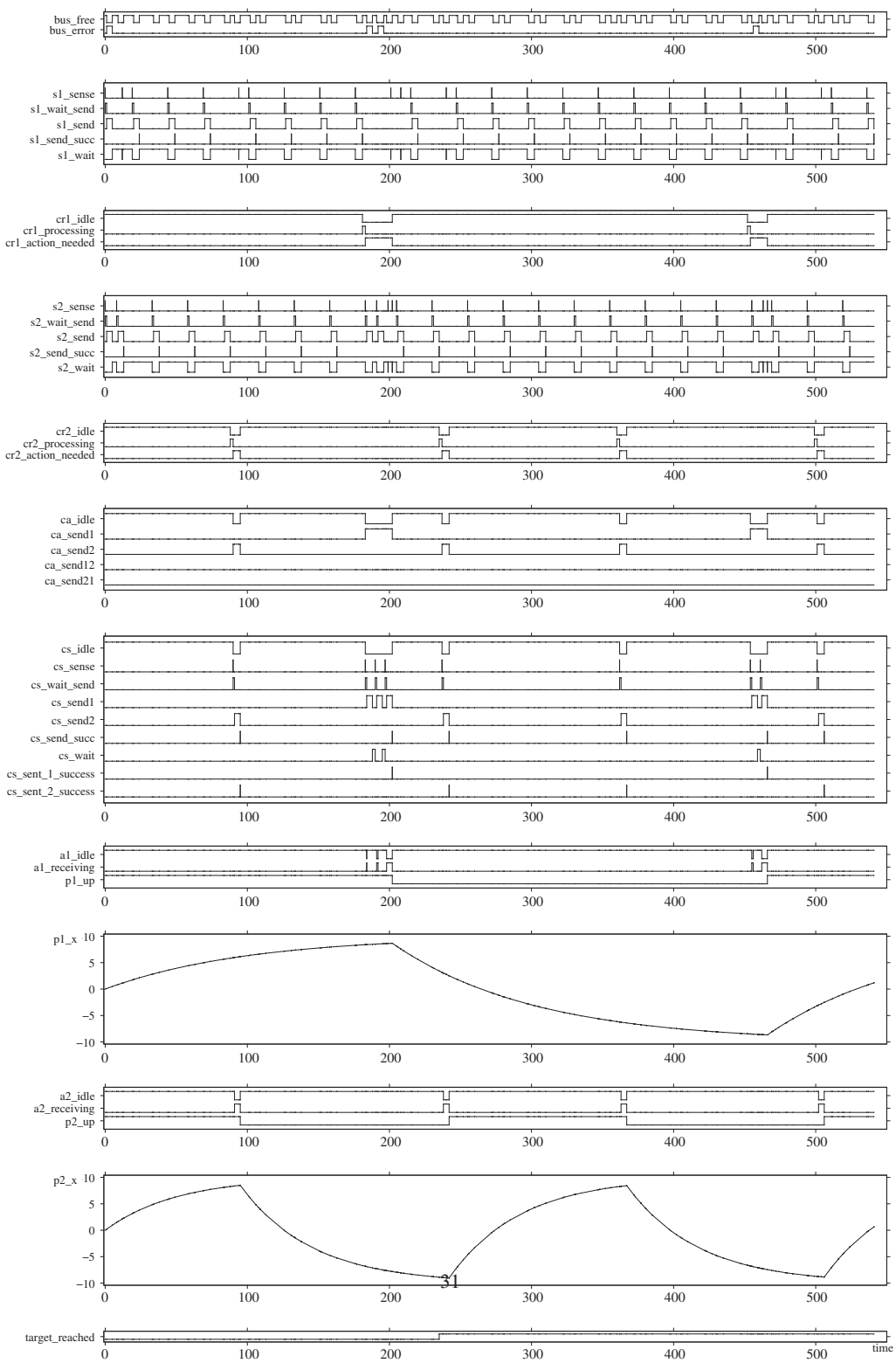




**Fig. 9**. Sample trace for the first scenario (i.e. without random quantification and fixed initial conditions) with direct ODE encoding. The time-axis is given by the actual time variable. The trace was calculated for 500 BMC steps, i.e. each graph consists of 500 sample points, which have been connected for illustration purposes only.

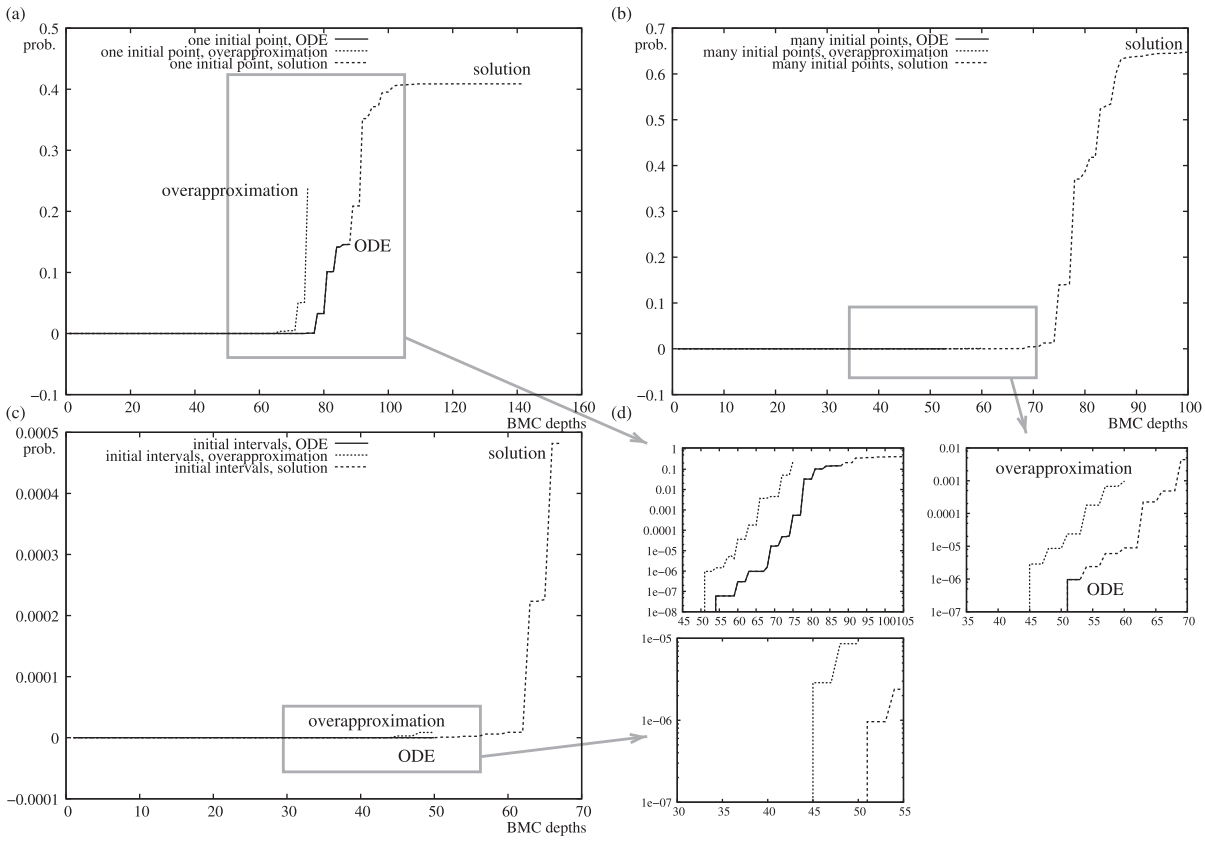


**Fig. 10**. Calculated maximum probabilities for the three different scenarios comprising random quantification: (a) scenario two with one fixed initial point, (b) scenario three with existential quantification of the initial point, (c) scenario four with existential quantification mapped to initial ranges. Subfigure (d) shows interesting excerpts in logarithmic scale to visualize the quickly growing gap between the overapproximation and the ODE/solution results.

3. The third scenario also uses random quantification but adds existential quantification for the initial values. Instead of setting them to fixed values, both $x_1$ and $x_2$ can be set to any value from $\{-2, -1, 0, 1, 2\}$ independently.
4. In the fourth scenario, a level of non-determinism for the initial values of $x1$ and $x_2$ is added. Instead of assigning the values from the existential quantifier's domain directly, they are used to map $x_1$ and $x_2$ to initial ranges from $\{[-2, -1], [-1, 0], [0, 1], [1, 2]\}$.

For each of these scenarios, the three different encodings of the continuous dynamics have been generated.

### *6.4. Analysis and results*

The goal of the controller is to keep the continuous variables inside the interval $[-8.8, 8.8]$. We thus are interested in the failure probability of the system, i.e. leaving this interval in at least one dimension. This goal is added to our encodings of the system as a description of the target states. The sample traces obtained from the first scenario show that this goal is reachable. Fig. 9 depicts one of those sample traces with a BMC unwinding depth of 500 for the encoding comprising ODEs. As can easily be seen from the figure, the trace clearly reflects the intended system behavior.

For the other three scenarios, we do not report single traces but the computed upper bounds of the probabilities to reach a target state, i.e. for $x_1$ or $x_2$ to become greater than 8.8 or less than $-8.8$. The experiments were performed in parallel on a 16 core 2.4 GHz AMD Opteron machine with 128 GB physical memory. The overall computation time for each of the 12 models (resulting from four scenarios and three encodings each) was limited to 1 week. We stopped computation at a BMC unwinding depth of 500 or after reaching that deadline.

For the three scenarios involving random quantification, Fig. 10 depicts the upper bounds on the worst-case probabilities resulting from the different encodings of the continuous behavior. It shows that the probabilities computed for the models incorporating ODEs and the models using the closed-form solutions coincide in all cases where the algorithms delivered analysis results within the given limit on runtime. Furthermore, the upper bounds on the probabilities obtained on the overapproximation are never tighter than those computed via ODE solving or closed-form solutions, i.e. they either coincide or are weaker in the sense of being strictly greater. The results even suggest that there is an exponentially growing gap (cf. Fig. 10 (d)) between the probabilities computed via the overapproximation and the ones calculated by ODE solving.

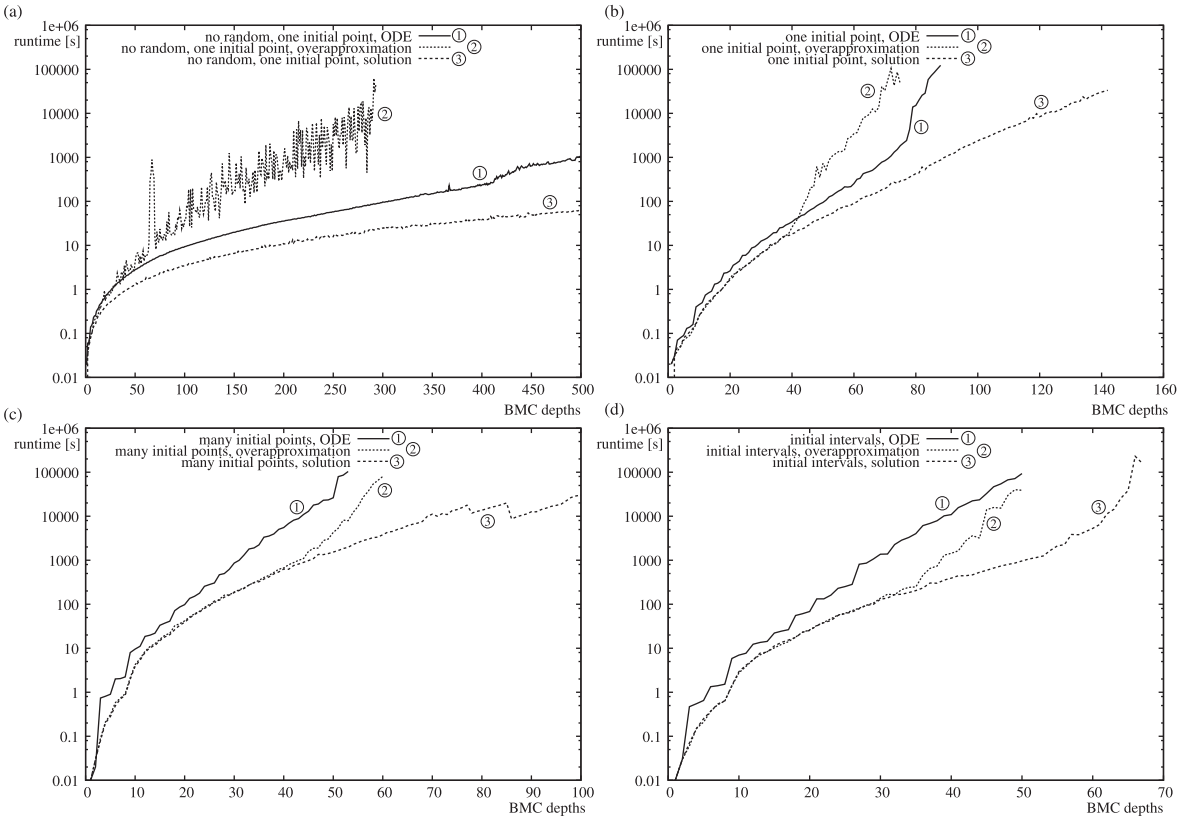


**Fig. 11**. Runtimes for the four scenarios of the case study: (a) scenario without random quantification and with fixed initial value, (b) scenario two with random quantification and fixed initial point, (c) scenario three with existential quantification of the initial point, (d) scenario four with existential quantification mapped to initial ranges.

The runtimes are shown in Fig. 11. Clearly, the solver performs best on the models making use of the exact solution, as expected. The runtimes for the overapproximating model stay very close to those of the exact solution for low unwinding depths, however increase dramatically later on. This is most likely due to the increased coarseness of the approximation, which causes a greater amount of non-determinism in the model. The solver thus has to cover a rapidly growing search space, e.g. whenever the overapproximated trajectories could either satisfy a transition guard or not, the solver has to investigate both options. Compared to the model encoding the exact solution, the overhead of using the embedded ODE solver is clearly noticeable. However, at least in the first two scenarios, at some point, the runtimes for the overapproximation exceed those for the ODE version and subsequently the ODE-enabled solver outperforms the one using the overapproximation. For the other two scenarios, a similar behavior is expected to occur eventually, as the runtime curves for the overapproximation version grow more steeply than the ones for the ODE version. The data collected within the runtime limitations is, however, insufficient to reveal an actual intersection between the respective curves in these scenarios.

Our experiments indicate that integrating ODE enclosure methods into the SiSAT tool is a promising approach. The combined algorithm using ODE enclosures outperforms the solver on a model using overapproximation of the continuous dynamics in terms of tightness of the calculated probabilities and in at least some of the scenarios even in terms of solver runtimes. Obviously, whenever the closed-form solution is readily available and expressible in the constraint language – with the latter not being very restrictive due to the rich set of arithmetic operators supported by SiSAT – one should make use of it. However, as closed-form solutions to differential equations are only available for a small subset of the ODEs encountered in models of technical systems, a solver that supports direct use of ODEs is much more versatile in practice.

## 7. Conclusion

This paper has given a detailed account of a symbolic encoding of probabilistic bounded reachability problems of dense-time probabilistic hybrid automata, together with a generalized SMT procedure determining the satisfaction probability of that encoding based on DPLL-style proof search and arithmetic constraint solving covering both non-linear arithmetics and ODEs. These ingredients provide a technique for analyzing probabilistic hybrid systems without resorting to approximation by intermediate finite-state abstractions and, due to the fully symbolic manipulation of discrete states facilitated by DPLL, without flattening out concurrent systems, thus potentially enhancing accuracy and scalability of the analysis algorithms.

The solving algorithm employs a tight integration of constraint solvers that has not been published before and which provides a completely new functionality. It is, however, built from components that have previously been implemented: An SMT-like procedure handling large Boolean combinations of non-linear arithmetic constraints, as encountered in the transition dynamics of general hybrid systems, has been implemented in the iSAT tool [32]. Its combination with safe enclosures for ODEs facilitating SMT reasoning over flows in hybrid systems has been prototypically implemented in [29]. Finally, a unification of SMT and stochastic SAT called SSMT has been explored in [61] and applied to discrete-time probabilistic hybrid automata in [33,62]. Our novel contribution here is to, first, unite those lines and, second, remove the severe semantic limitations of the original SSMT approach. This permits a generalization to dense-time probabilistic hybrid systems and non-deterministic assignments.

## Acknowledgments

We would like to thank our colleagues in the Transregional Research Center "AVACS" (Automatic Verification and Analysis of Complex Systems) for their contributions to the technologies underlying the combination of SMT techniques suggested herein. In particular, we would like to thank Holger Hermanns and Lijun Zhang for familiarizing us with probabilistic model checking in general and the intricacies of step-bounded probabilistic properties in particular, Stefan Ratschan for introducing us to interval constraint propagation and for discussing constraint-based reasoning over ODEs, and Christian Herde, Natalia Kalinnik, Stefan Kupferschmid, Tobias Schubert, and Bernd Becker for jointly developing and implementing the iSAT algorithm. Finally, we would like to thank the anonymous reviewers for their extremely helpful and exceptionally detailed comments.

## References

[1] E. Ábrahám, B. Becker, F. Klaedke, M. Steffen, Optimizing bounded model checking for linear hybrid systems, in: R. Cousot (Ed.), Proc. of Sixth Int. Conf. on Verification, Model Checking and Abstract Interpretation, VMCAI 2005, Paris, January 2005, Lect. Notes in Comput. Sci, vol. 3385, Springer, 2005, pp. 396–412.
[2] E. Ábrahám, T. Schubert, B. Becker, M. Fränzle, C. Herde, Parallel SAT solving in bounded model checking, J. Log. Comput., in press, doi:10.1093/logcom/exp002.
[3] S. Amin, A. Abate, M. Prandini, J. Lygeros, S. Sastry, Reachability analysis of controlled discrete-time stochastic hybrid systems, in: J.P. Hespanha, A. Tiwari (Eds.), Proc. of Ninth Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2006, Santa Barbara, CA, March 2006, Lecture Notes in Computer Science, vol. 3927, Springer, 2006, pp. 49–63.
[4] L. Arnold, Stochastic Differential Equations: Theory and Applications, Wiley, 1974.
[5] G. Audemard, M. Bozzano, A. Cimatti, R. Sebastiani, Verifying industrial hybrid systems with mathsat, in: A. Biere, O. Strichman (Eds.), Proc. of Second Int. Wksh. on Bounded Model-Checking, BMC (Boston, MA, July), Electron. Notes in Theor. Comput. Sci., vol. 119 (2) , Elsevier, 2005, pp. 17–32.
[6] C. Baier, H. Hermanns, J.-P. Katoen, B.R. Haverkort, Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes, Theor. Comput. Sci. 345 (1) (2005) 2–26.
[7] C. Barrett, R. Sebastiani, S.A. Seshia, C. Tinelli, Satisfiability modulo theories, in: A. Biere, M.J.H. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, 2009, pp. 825–885.
[8] A. Bauer, M. Pister, M. Tautschnig, Tool-support for the analysis of hybrid systems and models, in: Proc. of 2007 Conf. on Design, Automation and Test in Europe (Nice, April 2007), EDA Consortium, 2007, pp. 924–929.
[9] R.J. Bayardo, R. Schrag, Using CSP look-back techniques to solve real-world SAT instances, in: Proc. of 14th Nat. Conf. on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conf. AAAI/'97/IAAI '97 (Providence, RI, July 1997), AAAI Press/The MIT Press, 1997, pp. 203-208.
[10] R. Bellman, A Markovian decision process, J. Math. Mech. 6 (1957) 679–684.
[11] A. Bemporad, S.D. Cairano, Optimal control of discrete hybrid stochastic automata, in: M. Morari, L. Thiele (Eds.), Proc. of Eighth Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2005 (Zürich, March 2005), Lecture Notes in Computer Science, vol. 3414, Springer, 2005, pp. 151–167.
[12] F. Benhamou, Heterogeneous constraint solving, in: M. Hanus, M. Rodríguez-Artalejo (Eds.), Proc. of 5th Int. Conf. on Algebraic and Logic Programming, ALP '96 (Aachen, September 1996), Lecture Notes in Computer Science, vol. 1139, Springer, 1996, pp. 62–76.
[13] F. Benhamou, L. Granvilliers, Continuous and interval constraints, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Foundations of Artificial Intelligence, , Elsevier, 2006, pp. 571–603.
[14] F. Benhamou, D. McAllester, P. Van Hentenryck, CLP(intervals) revisited, in: M. Bruynooghe (Ed.), Proc. of 1994 Int. Symp. on Logic Programming, IPLS '94 (Ithaca, NY, November 1994), MIT Press, 1994, pp. 124–138.
[15] M. Berz, K. Makino, Verified integration of ODEs and flows using differential algebraic methods on high-order Taylor models, Reliab. Comput. 4 (4) (1998) 361–369.
[16] A. Biere, A. Cimatti, E. Clarke, Y. Zhu, Symbolic model checking without BDDs, in: R. Cleaveland (Ed.), Proc. of Fifth Int. Conf. on Tools for Construction and Analysis of Systems, TACAS '99 (Seattle, WA, Aug. 2006), Lecture Notes in Computer Science, vol. 4144, Springer, 2006, pp. 81–94.
[17] H.A. Blom, J. Lygeros (Eds.), Stochastic Hybrid Systems: Theory and Safety Critical Applications, Lecture Notes in Control and Information Sciences, vol. 337, Springer, 2006
[18] H.A.P. Blom, J. Krystul, G.J. Bakker, A particle system for safety verification of free flight in air traffic, Decision and Control, IEEE, 2006, pp. 1574–1579.
[19] M. Bozzano, R. Bruttomesso, A. Cimatti, T.A. Junttila, P. van Rossum, S. Schulz, R. Sebastiani, The MathSAT 3 system, in: R. Nieuwenhuis (Ed.), Proc. of 20th Int. Conf. on Automated Deduction CADE-20 (Tallinn, July 2005), Lecture Notes in Artificial Intelligence, vol. 3632, Springer, 2005, pp. 315–321.
[20] M.L. Bujorianu, J. Lygeros, Reachability questions in piecewise deterministic Markov processes, in: O. Maler, A. Pnueli (Eds.), Proc. of Sixth Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2003 (Prague, April 2003), Lecture Notes in Computer Science, vol. 2623, Springer, 2003, pp. 126–140.
[21] M.L. Bujorianu, J. Lygeros, Toward a general theory of stochastic hybrid systems, Stochastic Hybrid Systems: Theory and Safety Critical Applications, Lecture Notes in Control and Information Science, vol. 337, Springer, 2006, pp. 3–30.
[22] C.G. Cassandras, J. Lygeros (Eds.), Stochastic Hybrid Systems, CRC Taylor & Francis, 2007.
[23] A. Cimatti, M. Pistore, M. Roveri, R. Sebastiani, Improving the encoding of LTL model checking into SAT, in: A. Cortesi (Ed.), Revised Papers from 3rd Int. Wksh. on Verification, Model Checking and Abstract Interpretation VMCAI 2002 (Venice, Jan. 2002), Lecture Notes in Computer Science, vol. 2294, Springer, 2002, pp. 196–207.
[24] M.H.A. Davis, Piecewise-deterministic markov processes: a general class of non-diffusion stochastic models, J. Roy. Statist. Soc. 46 (3) (1984) 353–384.
[25] M. Davis, Markov Models and Optimization, Chapman and Hall, London, 1993.
[26] M. Davis, H. Putnam, A computing procedure for quantification theory, J. ACM 7 (3) (1960) 201–215.
[27] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Commun. ACM 5 (1962) 394–397.

[28] B. Dutertre, L. de Moura, A fast linear-arithmetic solver for DPLL(T), in: T. Ball, R.B. Jones (Eds.), Proc. of 18th Computer-Aided Verification Conference, CAV 2006 (Seattle, WA, Aug. 2006), Lecture Notes in Computer Science, vol. 4144, Springer, 2006, pp. 81–94.
[29] A. Eggers, M. Fränzle, C. Herde, SAT modulo ODE: a direct SAT approach to hybrid systems, in: S.S. Cha, J.-Y. Choi, M. Kim, I. Lee, M. Viswanathan (Eds.), Proc. of Sixth Int. Symp. on Automated Technology for Verification and Analysis, ATVA 2008 (Seoul, October 2008), ATVA, Lect. Notes in Comput. Sci, vol. 5311, Springer, 2008, pp. 171–185.
[30] X. Feng, K.A. Loparo, Y. Ji, H.J. Chizeck, Stochastic stability properties of jump linear systems, IEEE Trans. Autom. Control 37 (1) (1992) 38–53.
[31] M. Fränzle, C. Herde, HySAT: an efficient proof engine for bounded model checking of hybrid systems, Formal Methods Syst. Design 30 (3) (2007) 179–198.
[32] M. Fränzle, C. Herde, T. Teige, S. Ratschan, T. Schubert, Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure, J. Satisfiability, Boolean Model. Comput. 1 (2007) 209–236.
[33] M. Fränzle, H. Hermanns, T. Teige, Stochastic satisfiability modulo theory: a novel technique for the analysis of Probabilistic hybrid systems, in: M. Egerstedt, B. Mishra (Eds.), Proc. of 11th Int. Conf. on Hybrid Systems: Computation and Control, HSCC 2008 (St. Louis, MO, April 2008), Lecture Notes in Computer Science, vol. 4981, Springer, 2008, pp. 172–186.
[34] A. Girard, Reachability of uncertain linear systems using zonotopes, in: M. Morari, L. Thiele (Eds.), Proc. of Eighth Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2005 (Zürich, March 2005), Lecture Notes in Computer Science, vol. 3414, Springer, 2005, pp. 291–305.
[35] J. Greifeneder, J. Frey, Probabilistic hybrid automata with variable step width applied to the analysis of networked automation systems, in: Proc. of Third IFAC Wksh. on Discrete Event System Design, DESDes '06 (Rydzyna, September 2006), 2006, pp. 283–288.
[36] J.F. Groote, J.W.C. Koorn, S.F.M. van Vlijmen, The safety guaranteeing system at Station Hoorn-Kersenboogerd, in: Conf. on Computer Assurance, National Institute of Standards and Technology, 1995, pp. 57–68.
[37] E.C.R. Hehner, Predicative programming, Commun. ACM 27 (1984) 134–151.
[38] J.P. Hespanha, Polynomial stochastic hybrid systems, in: M. Morari, L. Thiele (Eds.), Proc. of Eighth Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2005 (Zürich, March 2005), Lecture Notes in Computer Science, vol. 3414, Springer, 2005, pp. 322–338.
[39] T.J. Hickey, D.K. Wittenberg, Rigorous modeling of hybrid systems using interval arithmetic constraints, in: R. Alur, G.J. Pappas (Eds.), Proc. of Seventh Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2004 (Philadelphia, PA, March 2004), Lecture Notes in Computer Science, vol. 2993, Springer, 2004, pp. 402–416.
[40] T. Hickey, Q. Ju, M. van Emden, Interval arithmetic: from principles to implementation, J. ACM 48 (5) (2001) 1038–1068.
[41] J. Hu, J. Lygeros, S. Sastry, Towards a theory of stochastic hybrid systems, in: N.A. Lynch, B.H. Krogh (Eds.), Proc. of Third Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2000 (Pittsburgh, PA, March 2000), Lecture Notes in Computer Science, vol. 1790, Springer, 2000, pp. 160–173.
[42] X.D. Koutsoukos, D. Riley, Computational methods for reachability analysis of stochastic hybrid systems, in: J.P. Hespanha, A. Tiwari (Eds.), Proc. of 9th Int. Wksh. on Hybrid Systems: Computation and Control, HSCC 2006 (Santa Barbara, CA, March 2006), Lecture Notes in Computer Science, vol. 3927, Springer, 2006, pp. 377–391.
[43] M.L. Littman, S.M. Majercik, T. Pitassi, Stochastic Boolean satisfiability, J. Autom. Reason. 27 (3) (2001) 251–296.
[44] R. Lohner, Einschließung der Lösung gewöhnlicher Anfangs- und Randwertaufgaben, PhD thesis, Univ. Karlsruhe, 1998.
[45] S. Majercik, Nonchronological backtracking in stochastic Boolean satisfiability, Proc. of 16th IEEE Int. Conf. on Tools with Artificial Intelligence, ICTAI 2004 (Boca Raton, FL, Nov. 2004), IEEE CS Press, 2004, pp. 498–507.
[46] S.M. Majercik, APPSSAT: approximate probabilistic planning using stochastic satisfiability, Int. J. Approx. Reason. 45 (2) (2007) 402–419.
[47] S.M. Majercik, Stochastic Boolean satisfiability, in: A. Biere, M.J.H. Heule, H. van Maaren, T. Walsh (Eds.), Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, IOS Press, 2009, pp. 887–925.
[48] S.M. Majercik, M.L. Littman, MAXPLAN: a new approach to probabilistic planning, in: R.G. Simmons, M.M. Veloso, S. Smith (Eds.), Proc. of 4th Int. Conf. Artificial Intelligence Planning Systems, AIPS '98 (Pittsburgh, PA), AAAI, 1998, pp. 86–93.
[49] K. Makino, M. Berz, Suppression of the wrapping effect by Taylor model-based verified integrators: long-term stabilization by preconditioning, Int. J. Differ. Equ. Appl. 10 (4) (2005) 353–384.
[50] K. Makino, M. Berz, Suppression of the wrapping effect by Taylor model-based verified integrators: the single step, Int. J. Pure Appl. Math. 36 (2) (2006) 175–197.
[51] R.E. Moore, Interval Analysis, Prentice Hall, 1966.
[52] R. Moore, Interval Analysis, Prentice-Hall, 1966.
[53] N.S. Nedialkov, Interval tools for ODEs and DAEs, Tech. Rep. CAS 06-09-NN, McMaster University, Hamilton, ON, 2006. Available online at <http://www.cas.mcmaster.ca/∼nedialk/PAPERS/intvtools/intvtools.pdf>.
[54] A. Neumaier, Interval Methods for Systems of Equations, Cambridge University Press, Cambridge, 1990.
[55] R. Nieuwenhuis, A. Oliveras, C. Tinelli, Solving SAT and SAT modulo theories: from an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL(T), J. ACM 53 (6) (2006) 937–977.
[56] C.H. Papadimitriou, Games against nature, J. Comput. Syst. Sci. 31 (2) (1985) 288–301.
[57] A. Pnueli, The temporal logic of programs, Proc. of 18th Ann. Symp. on Foundations of Computer Science, FOCS '77 (Providence, RI, October–November 1977), IEEE CS Press, 1977, pp. 46–57.
[58] D. Riley, X.D. Koutsoukos, K. Riley, Reachability analysis of a biodiesel production system using stochastic hybrid systems, in: Proc. of 15th IEEE Mediterranean Conf. on Control and Automation, MED '07 (Athens, June 2007), IEEE, 2007, 6 pp.
[59] J. Sproston, Model Checking of Probabilistic Timed and Hybrid Systems, PhD thesis, Univ. of Birmingham, 2000.
[60] J. Sproston, Decidable model checking of probabilistic hybrid automata, in: M. Joseph (Ed.), Proc. of Sixth Int. Symp. on Formal Techniques for Real-Time and Fault-Tolerant Systems, FTRTFT 2000 (Pune, Sept. 2000), Lecture Notes in Computer Science, vol. 1926, Springer, 2000, pp. 31–45.
[61] T. Teige, M. Fränzle, Stochastic satisfiability modulo theories for non-linear arithmetic, in: L. Perron, M.A. Trick (Eds.), Proc. of Fifth Int. Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, CPAIOR 2008 (Paris, May 2008), Lecture Notes in Computer Science, vol. 5015, Springer, 2008, pp. 248–262.
[62] T. Teige, M. Fränzle, Constraint-based analysis of probabilistic hybrid systems, in: A. Giua, C. Mahulea, M. Silva, J. Zaytoon (Eds.), Proc. of Third IFAC Conf. on Analysis and Design of Hybrid Systems, ADHS'09 (Zaragoza, Sept. 2009), IFAC, 2009, pp. 162–167.
[63] T. Walsh, Stochastic constraint programming, in: F. van Harmelen (Ed.), Proc. of the 15th Europ. Conf. on Artificial Intelligence, ECAI '02 (Lyon, July 2002), IOS Press, 2002, pp. 111–115.
[64] L. Zhang, C.F. Madigan, M.H. Moskewicz, S. Malik, Efficient conflict driven learning in a Boolean satisfiability solver, Proc. of 2001 IEEE/ACM Int. Conf. on Computer-Aided Design, ICCAD '01 (San Jose, CA, November 2001), IEEE, 2001, pp. 279–285.