Stochastic Satisfiability Modulo Theory: A Novel Technique for the Analysis of Probabilistic Hybrid Systems*

Martin Fränzle[1], Holger Hermanns[2], and Tino Teige[1]

[1] Carl von Ossietzky Universität, Oldenburg, Germany
{fraenzle|teige}@informatik.uni-oldenburg.de
[2] Saarland University, Saarbrücken, Germany
hermanns@cs.uni-sb.de

Abstract. The analysis of hybrid systems exhibiting probabilistic behaviour is notoriously difficult. To enable mechanised analysis of such systems, we extend the reasoning power of arithmetic satisfiability-modulo-theory solving (SMT) by a comprehensive treatment of randomized (a.k.a. stochastic) quantification over discrete variables within the mixed Boolean-arithmetic constraint system. This provides the technological basis for a fully symbolic analysis of probabilistic hybrid automata. Generalizing SMT-based bounded model-checking of hybrid automata [2, 11], stochastic SMT permits the direct and fully symbolic analysis of probabilistic bounded reachability problems of probabilistic hybrid automata without resorting to approximation by intermediate finite-state abstractions.

1 Introduction

Over the last decade, formal verification of digital systems has evolved from an academic subject to an approach accepted by industry, with dozens of commercial tools now available. Among the most successful verification methods for finite-state systems is *bounded model checking* (BMC), as suggested by Groote et al. in [13] and by Biere et al. in [3]. 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 tableau 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

* This work has been partially 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).

constraints over real-valued variables, thus entailing the need for satisfiability-modulo-theory (SMT) solvers over arithmetic theories to solve them. Such SMT procedures are thus currently in the focus of the SAT-solving community (e.g., [10]), as is their application to and tailoring for BMC of hybrid system (e.g., [2]).

The scope of these procedures, however, is confined 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 degree of continuous dynamics, the support for random phenomena, 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 experiments [20], *piecewise deterministic Markov processes* [8], where state changes may happen spontaneously in a manner similar to continuous-time Markov processes, and *stochastic differential equations* [1], 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 [16, 6]. While such models have a vast potential of application, results related to their analysis and verification are limited, and often based on Monte-Carlo simulation [4, 15]. For certain subclasses of piecewise deterministic Markov processes, of probabilistic hybrid automata, and of stochastic hybrid systems, reachability probabilities can be approximated (e.g. [20, 7, 17]).

In this paper, we present a technology that saves the virtues of SMT-based BMC, namely the fully symbolic treatment of hybrid state spaces, while advancing the reasoning power to probabilistic models and requirements. While the technique is more general, the current paper focuses on depth-bounded reachability of discrete-time probabilistic hybrid automata. With respect to the stochastic dynamics considered this model is very simple and thus constitutes a good attack point to pioneer effective model checking techniques for probabilistic hybrid systems, harvesting recent advances in depth-bounded reachability analysis for ordinary hybrid systems. Albeit being simple, the model of probabilistic hybrid automata has interesting practical applications [20].

In order to achieve this goal, we first define *stochastic satisfiability modulo theory* (SSMT) as the unification of stochastic propositional satisfiability [18] and satisfiability modulo theory. We proceed in Section 3 by defining discrete-time probabilistic hybrid automata. Section 4 formalizes the SSMT encoding of their probabilistic bounded reachability properties. Together with an extension of SMT solving to SSMT solving explained in Section 5, this symbolic encoding provides fully symbolic analysis of probabilistic bounded reachability problems of probabilistic hybrid automata without resorting to approximation by intermediate finite-state abstractions.

2 Stochastic satisfiability modulo theories

The *satisfiability modulo theories* (SMT) problem is a decision problem for logical formulae wrt. combinations of background theories. In this section we extend the SMT problem for arithmetic theories over the real numbers to support *randomized quantification* over discrete variables as known from *stochastic satisfiability* (SSAT) [18] and *stochastic constraint programming* (SCP) [5].

Let φ be an SMT formula in conjunctive normal form (CNF) over some quantifier-free arithmetic theory T. I.e., φ is a logical *conjunction* of clauses, and a *clause* is a logical *disjunction* of (atomic) arithmetic predicates from T, as in $\varphi = (x > 0 \vee 2a + 4b \geq 3) \wedge (y > 0 \vee 2a + 4b < 1)$. An SSMT 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 quantifier Q_i,[3] and an SMT formula φ, also called *matrix*. We require that the domains $\mathrm{dom}(x)$ of quantified variables x are finite (and thus discrete). A quantifier Q_i, associated with variable x_i, is either *existential*, denoted as $\exists$, or *randomized*, denoted as Я_{d_i} where d_i is a discrete probability distribution over $dom(x_i)$. The value of a variable x_i bound by a randomized quantifier (randomized variable for short) is determined stochastically by the corresponding distribution d_i, while the value of an existentially quantified variable can be set arbitrarily. We usually denote such a probability distribution d_i by a list $\langle (v_1, p_1), \ldots, (v_m, p_m) \rangle$ of value pairs, where p_j is understood as the probability of setting variable x_i to v_j. The list satisfies $v_j \neq v_k$ for $j \neq k$, $\forall j : p_j > 0$, $\sum_{j=1}^{m} p_j = 1$, and $\mathrm{dom}(x_i) = \{v_1, \ldots, v_m\}$. For instance, $\text{Я}_{\{(0,0.2),(1,0.5),(2,0.3)\}} x \in \{0, 1, 2\}$ means that the variable x is assigned the value 0, 1, or 2 with probability 0.2, 0.5, and 0.3, respectively.

The semantics of an SSMT problem is defined by the *maximum probability of satisfaction.* Intuitively, for an SSMT formula $\Phi = \exists x_1 \in \mathrm{dom}(x_1)\ \text{Я}_{d_2} x_2 \in \mathrm{dom}(x_2)\ \exists x_3 \in \mathrm{dom}(x_3)\ \text{Я}_{d_4} x_4 \in \mathrm{dom}(x_4) : \varphi$ determine the maximum probability s.t. there is a value for x_1 s.t. for random values of x_2 there is a value for x_3 s.t. for random values of x_4 the SMT formula φ is satisfiable. (As standard, an SMT formula φ (in CNF) is *satisfiable* iff there exists a valuation σ of the variables in φ s.t. each clause is satisfied under σ, i.e., iff at least one atom in each clause is satisfied under σ. Otherwise, φ is *unsatisfiable.*) More formally, the maximum probability of satisfaction $Pr(\Phi)$ of an SSMT formula Φ is defined recursively by the following rules where φ denotes the matrix.

1. $Pr(\varphi) \ = \ 0$ if φ is unsatisfiable, and 1 otherwise.
2. $Pr(\exists x_i \in \mathrm{dom}(x_i) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi)$
   $= \ \max_{v \in \mathrm{dom}(x_i)} Pr(Q_{i+1} x_{i+1} \in \mathrm{dom}(x_{i+1}) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi[v/x_i])$.

[3] Not all variables occurring in the formula φ need to be bound by a quantifier.

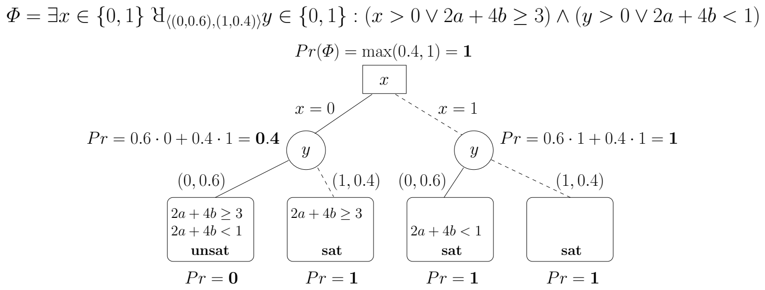
$\Phi = \exists x \in \{0,1\}\ \rotatebox[origin=c]{180}{R}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} : (x > 0 \lor 2a + 4b \geq 3) \land (y > 0 \lor 2a + 4b < 1)$
$Pr(\Phi) = \max(0.4, 1) = \mathbf{1}$
x
$x = 0$
$x = 1$
$Pr = 0.6 \cdot 0 + 0.4 \cdot 1 = \mathbf{0.4}$
y
y
$Pr = 0.6 \cdot 1 + 0.4 \cdot 1 = \mathbf{1}$
$(0, 0.6)$
$(1, 0.4)$
$(0, 0.6)$
$(1, 0.4)$
$2a + 4b \geq 3$
$2a + 4b < 1$
unsat
$2a + 4b \geq 3$
sat
$2a + 4b < 1$
sat
sat
$Pr = \mathbf{0}$
$Pr = \mathbf{1}$
$Pr = \mathbf{1}$
$Pr = \mathbf{1}$

Fig. 1. Semantics of an SSMT formula depicted as a tree.

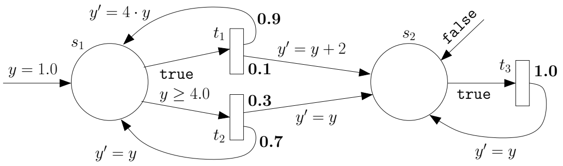

Fig. 2. A probabilistic hybrid automaton A.

3. $Pr(\rotatebox[origin=c]{180}{R}_{d_i} x_i \in \mathrm{dom}(x_i) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi)$
$= \sum_{(v,p) \in d_i} p \cdot Pr(Q_{i+1} x_{i+1} \in \mathrm{dom}(x_{i+1}) \ldots Q_n x_n \in \mathrm{dom}(x_n) : \varphi[v/x_i])$.

For an example see Fig. 1.

3 Probabilistic hybrid automata

A *discrete-time probabilistic hybrid automaton* $A = (\Sigma, \mathit{Trans}, R, s, p, g, \mathit{asgn}, \mathit{init})$, as depicted in Fig. 2, consists of

- finite sets Σ of *locations*, *Trans* of *transitions*, and $R = \{x_1, \ldots, x_n\}$ of *continuous state components*, together with mappings $s : \mathit{Trans} \to \Sigma$, assigning to each transition its source location, and $p : \mathit{Trans} \to P(\Sigma)$, assigning to each transition a probability distribution over the target locations,[4]
- 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 in our arithmetic theory T with free variables in R,
- a family $\mathit{asgn} = (\mathit{asgn}_{t,\sigma'})_{t \in \mathit{Trans}, \sigma' \in \Sigma}$ assigning to each transition and each target location an *assignment* which is defined by means of a T-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. We demand that assignments are uniquely defined for each state satisfying the guard, i.e. require
  1. *Definedness:* $g_t \Rightarrow \exists x'_1, \ldots, x'_n : \mathit{asgn}_{t,\sigma'}$ and

[4] W.l.og., distributions range over the full set Σ as unconnected locations and locations connected with probability 0 are indistinguishable wrt. probabilistic reachability.

2. *Determinacy:*
$$g_t \Rightarrow \forall x'_1,\dots,x'_n, y'_1,\dots,y'_n : \left(\begin{array}{l} asgn_{t,\sigma'} \land asgn_{t,\sigma'}[y'_1,\dots,y'_n/x'_1,\dots,x'_n] \\ \Rightarrow \forall i \le n : x'_i = y'_i \end{array}\right)$$

to be valid for each $t \in \mathit{Trans}$ and $\sigma' \in \Sigma$.

- a family $\mathit{init} = (\mathit{init}_\sigma)_{\sigma\in\Sigma}$ of *initial state predicates*, where each init_σ is a T-predicate over R which constrains the valuations of the continuous state components when control resides *initially* in the discrete location σ.[5] For technical reasons, we demand that for each $\sigma \in \Sigma$, there is at most one $\boldsymbol{x} \in R \to \mathbb{R}$ which satisfies the predicate init_σ.

The automaton engages in a sequence of steps coinciding to its transitions, thereby selecting among the enabled transitions and assigning a sequence of valuations to the continuous variables which is consistent with the transition effects. A *step* can be represented by a tuple $(\sigma, \boldsymbol{x}, t, \sigma', \boldsymbol{x}')$ consisting of a source location $\sigma \in \Sigma$ and a target location σ', a continuous source state $\boldsymbol{x} \in (R \to \mathbb{R})$ and a continuous target state $\boldsymbol{x}' \in (R \to \mathbb{R})$, plus a transition $t \in \mathit{Trans}$. Such a tuple is a step of automaton A iff there is a transition $t \in \mathit{Trans}$ with $s(t) = \sigma$ such that $\boldsymbol{x}$ satisfies g_t and such that $\mathit{asgn}_{t,\sigma'}$ is satisfied if $\boldsymbol{x}$ is substituted for the variables in R and $\boldsymbol{x}'$ is substituted for the variables in R'. Slightly abusing notation, we will denote the latter fact by $\boldsymbol{x}, \boldsymbol{x}' \models \mathit{asgn}_{t,\sigma'}$ in the sequel. A *run* of A is an alternating sequence $r = (\sigma_0, \boldsymbol{x}_0) \xrightarrow{t_0}, \dots, \xrightarrow{t_{n-1}} (\sigma_n, \boldsymbol{x}_n)$ of *hybrid states* $(\sigma_i, x_i) \in \Sigma \times (R \to \mathbb{R})$ and transitions $t_i \in \mathit{Trans}$, built from steps of A grounded in a viable initial state. I.e., $\boldsymbol{x}_0$ satisfies init_{σ_0} and for all $i < n$, the tuple $(\sigma_i, \boldsymbol{x}_i, t_i, \sigma_{i+1}, \boldsymbol{x}_{i+1})$ is a step of A.

In the sequel, we will be interested in the probability of reaching a given set of undesirable locations within a given number of steps. Owed to the presence of nondeterminism, this probability measure is well-defined only if considering a particular policy (scheduler, adversary) that resolves the nondeterminism. We are interested in the *worst-case*, i.e., *maximum probability of reaching the undesirable states* achieved if ranging over arbitrary policies that may resolve nondeterminism using randomization, the history, etc. Since we are considering step-bounded probabilities, we can avoid the explicit introduction of policies, and instead define the probability of reaching some target state in a set TL of discrete locations within k steps directly as follows.

Definition 1 (Probabilistic bounded reachability). *Given a probabilistic hybrid automaton $A = (\Sigma, \mathit{Trans}, R, s, p, g, \mathit{asgn}, \mathit{init})$, a set of target locations $TL \subset \Sigma$, a depth $k \in \mathbb{N}$, and a hybrid state $(\sigma, \boldsymbol{x}) \in \Sigma \times (R \to \mathbb{R})$, the* maximum probability of reaching the target TL from $(\sigma, \boldsymbol{x})$ in at most k steps *is denoted $P^k_A(\sigma, \boldsymbol{x}, TL)$. It is defined recursively over the depth k as follows:*

$$P^k_A(\sigma, \boldsymbol{x}, TL) = \begin{cases} 1 & \text{if } \sigma \in TL, \\ 0 & \text{if } \sigma \notin TL \land k = 0, \\ \max_{t\in \mathit{Enabled}} \sum_{\sigma'\in\Sigma} \left(p(t)(\sigma') \cdot P^{k-1}_{t,\sigma'}\right) & \text{if } \sigma \notin TL \land k > 0, \end{cases}$$

[5] A discrete location σ not to be taken initially takes the predicate $\mathit{init}_\sigma = \texttt{false}$.

where $Enabled = \{t \in Trans, s(t) = \sigma, \boldsymbol{x} \models g(t)\}$ *and* $P^k_{t,\sigma'} = P^k_A(\sigma', \boldsymbol{x'}, TL)$ *for the unique* $\boldsymbol{x'}$ *with* $\boldsymbol{x}, \boldsymbol{x'} \models asgn_{t,\sigma'}$.

For an illustration of probabilistic bounded reachability consider the probabilistic hybrid automaton A from Fig. 2 with $TL = \{s_2\}$. Then $P^0_A(s_1, y \mapsto 1.0, TL) = 0.0$, $P^1_A(s_1, y \mapsto 1.0, TL) = 0.1$, $P^2_A(s_1, y \mapsto 1.0, TL) = 0.1 + 0.9 \cdot 0.3 = 0.37$, $P^3_A(s_1, y \mapsto 1.0, TL) = 0.1 + 0.9 \cdot (0.3 + 0.7 \cdot 0.3) = 0.559$.

Based on the worst-case probability of reaching TL, we define the *probabilistic bounded model checking problem (PBMC, for short)* to be the problem of deciding whether the maximum probability of reaching the undesirable states from an initial state within a given number of steps lies below a given threshold:

Definition 2 (Probabilistic bounded model checking). *Given a probabilistic hybrid automaton* $A = (\Sigma, Trans, R, s, p, g, asgn, init)$*, a set of target locations* $TL \subset \Sigma$*, a depth* $k \in \mathbb{N}$*, and a probability threshold* $x \in [0, 1]$*, the* probabilistic bounded model checking problem wrt. target states TL and depth k *is to determine whether* $\max\{P^k_A(\sigma, \boldsymbol{x}, TL) \mid \sigma \in \Sigma, \boldsymbol{x} \models init_\sigma\} \le x$.

4 Reducing PBMC to SSMT

In order to perform probabilistic bounded model checking (PBMC) we employ a reduction to stochastic satisfiability modulo theory (SSMT) which generalizes the propositional SAT encodings for bounded model checking of finite-state systems [3] and the SMT encodings for BMC of hybrid automata [2, 11]. 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 $k \in \mathbb{N}$, akin to [2, 11]. 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 $A = (\Sigma, Trans, R, s, p, g, asgn, init)$ be a discrete-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:

1. For encoding the discrete state $\sigma \in \Sigma$, we take $k+1$ variables σ^i, for $0 \le i \le k$, each with domain Σ. The value of σ^i coincides with the discrete location which automaton A resides in during step i.

2. For representing transitions $t \in Trans$, we take k variables t^i with domain $Trans$, for $1 \le i \le k$. The value of t^i encodes the ith move in the run of A.

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 ith transition in the run (and thus x^i the value thereafter).

4. The interplay between discrete states and transitions requires that t^i implies $\sigma^{i-1} = s(t^i)$. This can be expressed by the $k \cdot |Trans|$ SSMT clauses in

$$\bigwedge_{i=1}^{k} \bigwedge_{t \in Trans} \left(t^i = t \Rightarrow \sigma^{i-1} = s(t)\right) \quad ,$$

where $\varphi \Rightarrow \psi$ abbreviates $\neg\varphi \vee \psi$.

5. Similarly, enabledness of the transition, i.e. validity of the *transition guard* in the pre-state, is enforced through the constraint system

$$\bigwedge_{i=1}^{k} \bigwedge_{t \in \mathit{Trans}} \left(t^i = t \Rightarrow g_t[x_1^{i-1}, \ldots, x_n^{i-1}/x_1, \ldots, x_n]\right) \ ,$$

where $\{x_1, \ldots, x_n\} = R$. Since g_t need not be a simple T-constraint, the above formula is not necessarily in conjunctive normal form and thus not an SSMT matrix formula. An equisatisfiable CNF can, however, always be obtained by introduction of auxiliary variables as in [12].

6. Likewise, *assignments* are dealt with by

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

Due to the probabilistic choice between variants of the selected transition, the assignment depends on both the transition t^i and the actual target location σ^i.

7. Finally, we complete the matrix by adding constraints describing the allowable initial states through the SSMT constraint system $\bigwedge_{\sigma \in \Sigma} \left(\sigma^0 = \sigma \Rightarrow \mathit{init}_\sigma\right)$.

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 [11]. As in BMC [3], satisfaction of temporal properties on all runs of depth k can thus be checked by adding to the formula the k-fold unrolling of a tableaux of the (negated) property, then checking the resulting formula for unsatisfiability. Using standard techniques from predicative semantics [14], 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.

Phase 2: Encoding choices. Let φ 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:

8. An SSMT formula $\psi = \psi_1$ encoding the probabilitsic 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 first select a finite set $O = \{o_1, \ldots, o_n\}$ of choice options for randomized choices, a probability distribution $p_O : O \to (0, 1]$ over O, and a function $\mathit{pd} : \mathit{Trans} \times \Sigma \to 2^O$ such that these together satisfy

$$\begin{array}{l} \forall t \in \mathit{Trans}, \sigma \in \Sigma : \sum_{pc \in \mathit{pd}(t,\sigma)} p_O(pc) = p(t)(\sigma) \ \text{ and} \\ \forall t \in \mathit{Trans}, \sigma_1, \sigma_2 \in \Sigma : \mathit{pd}(t, \sigma_1) \cap \mathit{pd}(t, \sigma_2) = \emptyset \ . \end{array}$$

Such a set O and probability distribution p_O do always exist. The worst-case cardinality of O is the number $|\{p(t)(\sigma) \mid t \in \mathit{Trans}, \sigma \in \Sigma\}|$ 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 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 σ_i by means of the mapping *pd*. Therefore, ψ_i is defined recursively as follows: for $1 \leq i < k$,

$$\psi_i = \exists t^i \in \mathit{Trans} : \text{ꓤ}_{\langle (o_1, p_O(o_1)), \ldots, (o_n, p_O(o_n)) \rangle} pc_i \in O : \psi_{i+1} \ \text{, and}$$

$$\psi_k = \varphi \wedge \bigwedge_{k=1}^{n} \bigwedge_{t \in \mathit{Trans}} \bigwedge_{\sigma \in \Sigma} [(t_i = t \wedge \sigma_i = \sigma) \Rightarrow \bigvee_{o \in pd(t,\sigma)} pc_i = o] \ .$$

9. In order to solve the PBMC problem, it remains to choose the initial state maximizing the probability. This can be accomplished by existential quantification, yielding the formula $\mathit{PBMC}^k_{A,TL} = \exists \sigma^0 \in \Sigma : \psi$. Given the structural similarity between probabilistic bounded reachability and quantification in SSMT, this reduction is correct in the following sense:

Proposition 1 (Correctness of reduction). *$Pr(\mathit{PBMC}^k_{A,TL}) \leq x$ iff A satisfies the PBMC problem wrt. threshold x, depth k, and target states TL.*

5 Algorithm for SSMT

In this section we present our algorithm for calculating the maximum probability of satisfaction of an SSMT formula. More precisely, for a given SSMT formula Φ and a lower and upper threshold $t_l, t_u \in [0, 1]$ with $t_l \leq t_u$, the algorithm returns a witness value $p \leq Pr(\Phi)$ s.t. $p > t_u$ iff $Pr(\Phi) > t_u$, a value $p < t_l$ iff $Pr(\Phi) < t_l$, or otherwise (i.e., $t_l \leq Pr(\Phi) \leq t_u$) the value $p = Pr(\Phi)$. It computes the exact value of $Pr(\Phi)$ when taking $t_l = 0$ and $t_u = 1$, but whenever we are interested in a particular target probability x, it saves computational effort by not being forced to be exact about probabilities different from $x = t_l = t_u$. As a proof procedure, we generalize to SSMT the extended Davis-Putnam-Logemann-Loveland (DPLL) algorithm [9] for SSAT described in [18].

Our SSMT algorithm consists of *three layers*. The lowermost layer is a *theory solver* TS for reasoning about a conjunctive system over theory T. As the middle layer, an *SMT solver* for disjunctive systems over T employs the theory solver TS. Finally, the *SSMT solver* is an extension of the SMT layer to deal with existential and randomized quantification.

Theory layer. As in SAT-modulo-theory solving, the theory solver TS decides whether a conjunctive system M of atomic predicates from T is satisfiable over T. Furthermore, we support theory solvers which can deduce new information from the given facts in the form of forward inference, but do not require this functionality to be present or even complete. We denote these capabilities of the theory solver, which following the SMT tradition we assume as given, by three deduction rules:

$M \longrightarrow_{TS}$ `sat` iff M is satisfiable.
$M \longrightarrow_{TS}$ `unsat` iff M is unsatisfiable.
$M \longrightarrow_{TS} M \cdot \langle a \rangle$ only if M is satisfiable and $M \models a$.

SMT layer. The SMT layer is described by the following rules. For more details we refer the reader to, e.g., the survey in [19]. In the sequel, let φ be an SMT formula over T in CNF. A *choice* asserts a theory atom occurring in φ or its negation to enforce progress in the backtracking search (rule (1)).

$$\frac{a \in c \in \varphi, b = a \text{ or } b = \neg a, b \notin M}{(\varphi, M) \longrightarrow_{SMT} (\varphi, M \cdot \langle |, b \rangle)} \tag{1}$$

Here, M denotes the list of all asserted atoms and $a \in c \in \varphi$ means that there is a theory atom a occurring in some clause c of φ. In order to facilitate (non-chronological) backtracking, in addition, we intersperse a special marker symbol $|$ into M. When we consider M as a conjunctive system (for the theory solver TS) we neglect the marker symbols $|$ in M.

The rules (2) (*unit propagation*) and (3) (*theory propagation*) are applied if new facts can be deduced. The deduced atoms are added to M.

$$\frac{(a_1 \vee \ldots \vee a_m) \in \varphi, a_i \notin M, \neg a_i \notin M, \forall j \in \mathbb{N}_{\leq m} \text{ with } j \neq i : \neg a_j \in M}{(\varphi, M) \longrightarrow_{SMT} (\varphi, M \cdot \langle a_i \rangle)} \tag{2}$$

$$\frac{M \longrightarrow_{TS} M \cdot \langle a \rangle, a \notin M, \exists c \in \varphi : a \in c \text{ or } \neg a \in c}{(\varphi, M) \longrightarrow_{SMT} (\varphi, M \cdot \langle a \rangle)} \tag{3}$$

If a conflict occurs, i.e. the list of asserted atoms has become infeasible, an (small or even minimal) reason for this conflicting situation can be extracted. To prevent the solver from revisiting the same or a similar conflict, this information can be encoded as an additional, implied clause, a.k.a. *conflict clause*, and added to the formula. This is referred to as *conflict-driven clause learning* (rule (4)).

$$\frac{M = M' \cdot \langle | \rangle \cdot M'', a_1, \ldots, a_{k-1} \in M', a_k \in M'', \langle a_1, \ldots, a_k \rangle \longrightarrow_{TS} \texttt{unsat}}{(\varphi, M) \longrightarrow_{SMT} (\varphi \wedge (\neg a_1 \vee \ldots \vee \neg a_k), M)} \tag{4}$$

Note that there are many different techniques for (efficient) generation of such an infeasible subsystem $a_1, \ldots, a_k$ of M. In rule (4), we use the *unique implication point technique* in order to enforce progress upon non-chronological backtracking in rule (5) (cf., e.g., [19]).

To resolve the conflict, the solver *non-chronologically backtracks* to a previous node in the search tree while often skipping multiple nodes in the tree. The backtrack node can be computed by means of the conflict clause as given by rule (5). Due to the use of unique implication points in conflict clauses, we can enforce a conflict clause to become unit upon backtracking and do directly assert the propagated atom.

$$\frac{M = M' \cdot \langle | \rangle \cdot M'', c \in \varphi, a \in c, \neg a \in M'', \forall a' \in c \text{ with } a' \neq a : \neg a' \in M'}{(\varphi, M) \longrightarrow_{SMT} (\varphi, M' \cdot \langle a \rangle)} \tag{5}$$

If each clause in φ contains at least one asserted atom and the conjunction of all asserted atoms is satisfiable then the SMT formula is satisfiable.

$$\frac{M \longrightarrow_{TS} \texttt{sat}, \forall c \in \varphi \ \exists a \in c : a \in M}{(\varphi, M) \longrightarrow_{SMT} \texttt{sat}} \tag{6}$$

If a conflict cannot be resolved, i.e. there is no choice point to be revoked, the formula is unsatisfiable.

$$\frac{| \notin M, M \longrightarrow_{TS} \texttt{unsat}}{(\varphi, M) \longrightarrow_{SMT} \texttt{unsat}} \tag{7}$$

SSMT layer. Given the rules of the SMT layer, we construct the SSMT algorithm. Let $\Phi = Pre : \varphi$ be an SSMT formula. W.l.o.g., we assume that all possible value assignments for quantified variables of the SSMT formula Φ are encoded as clauses in the matrix of Φ. More formally, forall $(Qx \in \{v_1, \ldots, v_k\}) \in Pre$ there is a clause $(x = v_1 \vee \ldots \vee x = v_k) \in \varphi$. By this information, the domain emptiness of a quantified variable will be detected by the SMT solver. An SSMT deduction starts from a state $(Pre, \varphi, M, t_l, t_u)$, where $\Phi := Pre : \varphi$ is an SSMT formula, M is a list of asserted atoms, and t_l, t_u are the lower and upper target thresholds. The deduction yields either a new proof state of the same structure or a pair (p, φ') of a satisfaction probability and a new matrix, in which case the deduction terminates. If $(Pre, \varphi, M, t_l, t_u) \longrightarrow^*_{SSMT} (p, \varphi')$ then $p > t_u$ iff $Pr(\Phi) > t_u$ under M, i.e. iff $Pr(Pre : (\varphi \wedge M)) > t_u$, and analogously $p < t_l$ iff $Pr(\Phi) < t_l$ under M, and otherwise $p = Pr(\Phi)$ under M. The new matrix $\varphi' \supseteq \varphi$ potentially contains learned clauses, i.e. $\forall c \in \varphi' - \varphi : (\varphi \models c)$.

The SSMT layer consists of the following rules. To deal with quantified variables, we *branch* the search by assigning values and combine the results according to the semantics of Section 2 (rules (8) and (9)). For the branching SSMT calls we update the target thresholds correspondingly. I.e., in case of an existentially quantified variable we transmit t_l, t_u for the branch $x = v$ and $\max(t_l, p_1), t_u$ for the remaining subtree, since we can neglect probabilities of the remaining subtree less than the already computed value p_1 for branch $x = v$. For randomized variables, we take the probability p_v for the value v and the maximum possible remaining probability $p_r = \sum_{v' \in D - \{v\}, (v', p_{v'}) \in d} p_{v'}$ for all other values $v' \neq v$ into account. I.e., the lower and upper target thresholds for the call where x is assigned to v are $(t_l - p_r)/p_v$ and t_u/p_v, resp., since if $t_l - p_r$ cannot be reached by branch $x = v$ then t_l cannot be reached at all. If t_u is already exceeded by branch $x = v$ then we already exceeded the upper target threshold, which we later will exploit within pruning rules wrt. the threshold parameters. The target thresholds for all remaining branches decrease by the computed probability $p_v \cdot p_1$ for $x = v$.

$$\frac{\begin{array}{c}|D| \geq 2, v \in D,\\ (Pre, \varphi, M \cdot \langle x = v \rangle, t_l, t_u) \longrightarrow^*_{SSMT} (p_1, \varphi_1), consistent(\varphi_1, M),\\ (\exists x \in D \setminus \{v\} \cdot Pre, \varphi_1, M \cdot \langle x \neq v \rangle, \max(t_l, p_1), t_u) \longrightarrow^*_{SSMT} (p_2, \varphi_2)\end{array}}{(\exists x \in D \cdot Pre, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (\max(p_1, p_2), \varphi_2)} \tag{8}$$

where $consistent(\varphi_1, M) := (\neg \exists c \in \varphi_1 : \forall a \in c : M \cdot \langle a \rangle \longrightarrow_{TS} \texttt{unsat})$ indicates whether the new matrix φ_1 is consistent with the list M of asserted atoms.

(9)

$$\frac{\begin{array}{c}|D| \geq 2, v \in D, (v, p_v) \in d, p_r = \sum_{v' \in D - \{v\}, (v', p_{v'}) \in d} p_{v'},\\ (Pre, \varphi, M \cdot \langle x = v \rangle, (t_l - p_r)/p_v, t_u/p_v) \longrightarrow^*_{SSMT} (p_1, \varphi_1), consistent(\varphi_1, M),\\ (\rotatebox[origin=c]{180}{R}_d x \in D \setminus \{v\} \cdot Pre, \varphi_1, M \cdot \langle x \neq v \rangle, t_l - p_v \cdot p_1, t_u - p_v \cdot p_1) \longrightarrow^*_{SSMT} (p_2, \varphi_2)\end{array}}{(\rotatebox[origin=c]{180}{R}_d x \in D \cdot Pre, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p_v \cdot p_1 + p_2, \varphi_2)}$$

If it turns out that the upper target threshold t_u is already reached by the branch under investigation, then we can save visiting any other branch and instead return the positive result immediately via rules (10) and (11). If the remaining branches have insufficient probability mass to reach the lower target threshold

t_l then we return the negative result by rule (12) without further exploration. These pruning rules generalize the *thresholding* rules for the propositional case from [18].

$$(10)\qquad \frac{\begin{array}{c} v \in D, (\mathit{Pre}, \varphi, M \cdot \langle x = v\rangle, t_l, t_u) \longrightarrow^{*}_{SSMT} (p, \varphi'), \\ \mathit{consistent}(\varphi', M), p > t_u \text{ or } |D| = 1 \end{array}}{(\exists x \in D \cdot \mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p, \varphi')}$$

$$(11)\qquad \frac{\begin{array}{c} v \in D, (v, p_v) \in d, p_r = 0 + \sum_{v' \in D - \{v\}, (v', p_{v'}) \in d} p_{v'}, \\ (\mathit{Pre}, \varphi, M \cdot \langle x = v\rangle, (t_l - p_r)/p_v, t_u/p_v) \longrightarrow^{*}_{SSMT} (p, \varphi'), \\ \mathit{consistent}(\varphi', M), p_v \cdot p > t_u \text{ or } |D| = 1 \end{array}}{(\text{Я}_d x \in D \cdot \mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p_v \cdot p, \varphi')}$$

$$(12)\qquad \frac{\begin{array}{c} |D| \geq 2, v \in D, (v, p_v) \in d, p_r = \sum_{v' \in D - \{v\}, (v', p_{v'}) \in d} p_{v'}, \\ (\mathit{Pre}, \varphi, M \cdot \langle x = v\rangle, (t_l - p_r)/p_v, t_u/p_v) \longrightarrow^{*}_{SSMT} (p, \varphi'), \\ \mathit{consistent}(\varphi', M), t_l - p_v \cdot p > p_r \end{array}}{(\text{Я}_d x \in D \cdot \mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p_v \cdot p, \varphi')}$$

It could also happen that after the first SSMT call (for $x = v$), all remaining branches for x lead to probability 0. This is indicated by the returned new matrix φ', in particular by some learned conflict clause in φ', which is inconsistent under the list M of asserted atoms (without assignment $x = v$). In this case, rules (13) and (14) save unnecessary visits of the remaining branches.

$$(13)\qquad \frac{v \in D, (\mathit{Pre}, \varphi, M \cdot \langle x = v\rangle, t_l, t_u) \longrightarrow^{*}_{SSMT} (p, \varphi'), \mathit{inconsistent}(\varphi', M)}{(\exists x \in D \cdot \mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p, \varphi')}$$

$$(14)\qquad \frac{\begin{array}{c} v \in D, (v, p_v) \in d, p_r = 0 + \sum_{v' \in D - \{v\}, (v', p_{v'}) \in d} p_{v'}, \\ (\mathit{Pre}, \varphi, M \cdot \langle x = v\rangle, (t_l - p_r)/p_v, t_u/p_v) \longrightarrow^{*}_{SSMT} (p, \varphi'), \\ \mathit{inconsistent}(\varphi', M) \end{array}}{(\text{Я}_d x \in D \cdot \mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (p_v \cdot p, \varphi')}$$

where $\mathit{inconsistent}(\varphi', M) := \neg\mathit{consistent}(\varphi', M)$ means that at least one clause in the new matrix φ' is inconsistent with M which forces the solver to backtrack to a previous level (thereby avoiding computation of all other possible branches of x) while keeping the already calculated probability. Note that chained executions of that rule, which occur if the returned matrix φ' 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 all quantified variables have a definite value, i.e. the current prefix is empty, we can execute the *choice* rule (15).

$$(15)\qquad \frac{(\varphi, M) \longrightarrow_{SMT} (\varphi, M \cdot \langle |, a\rangle)}{(\emptyset, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (\emptyset, \varphi, M \cdot \langle |, a\rangle, t_l, t_u)}$$

If the SMT solver can propagate new facts from the matrix φ and the list M of asserted atoms, we can do the same in the SSMT layer. I.e., both *unit propagation* and *theory propagation* are lifted to SSMT by rule (16).

$$(16)\qquad \frac{(\varphi, M) \longrightarrow_{SMT} (\varphi, M \cdot \langle a\rangle)}{(\mathit{Pre}, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (\mathit{update}(\mathit{Pre}, a), \varphi, M \cdot \langle a\rangle, t_l, t_u)}$$

where *update*(Pre, a) prunes the domains $\mathrm{dom}(x)$ in the prefix Pre of the quantified variables x corresponding to the theory atom a. We do not require that *update* is a complete (yet a sound) pruning procedure, i.e., potentially not all but only some non-solutions are removed from $\mathrm{dom}(x)$. E.g., if $a = x > 3$ then the updated domain of x is $\{v_x \in \mathrm{dom}(x) : v_x > 3\}$ or a superset thereof in case of incomplete pruning.

A conflict clause learned by the SMT solver is also valid within the SSMT framework, i.e. *conflict-driven clause learning* is supported by rule (17). Note that implied clauses can be added at any point in the search, in particular even if the current prefix is non-empty.

$$(17) \qquad \frac{(\varphi, M) \longrightarrow_{SMT} (\varphi \wedge c, M)}{(Pre, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (Pre, \varphi \wedge c, M, t_l, t_u)}$$

Rule (18) enables the SSMT solver to *backjump* within the theory part of the search tree.

$$(18) \qquad \frac{(\varphi, M \cdot \langle | \rangle \cdot M') \longrightarrow_{SMT} (\varphi, M)}{(\emptyset, \varphi, M \cdot \langle | \rangle \cdot M', t_l, t_u) \longrightarrow_{SSMT} (\emptyset, \varphi, M, t_l, t_u)}$$

Since a marker symbol $|$ is added to the list of asserted atoms only if there is an empty prefix (cf. rule 15), i.e. if all quantified variables are assigned to a value, rule (18) guarantees that value assignments of quantified variables will *not* be removed from the list of asserted atoms (i.e., M' does not contain such assignments).

If a solution of the matrix φ is found by the SMT layer and the prefix Pre does *not* contain randomized quantifiers then the probability 1 is returned by rule (19), since for the remaining existentially quantified variables in Pre there is at least one satisfying branch. However, if some randomized quantifiers are included in Pre, we may not return the probability 1 for the entire subtree, since the initial domain for some randomized variables could be pruned (by rule (16)), and potentially the probability of satisfaction for that subtree could be less 1.

$$(19) \qquad \frac{(\varphi, M) \longrightarrow_{SMT} \texttt{sat}, (\reflectbox{R}_d x \in D) \notin Pre}{(Pre, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (1, \varphi)}$$

If the SMT solver finds out that the matrix φ is unsatisfiable under M then rule (20) may return the satisfaction probability 0 even if the prefix Pre is non-empty, since no assignment to the quantified variables could counterfeit the unsatisfiability of φ under M.

$$(20) \qquad \frac{(\varphi, M) \longrightarrow_{SMT} \texttt{unsat}}{(Pre, \varphi, M, t_l, t_u) \longrightarrow_{SSMT} (0, \varphi)}$$

The above unification of DPLL-based SSAT solving with of SMT is sound and complete in the following sense.

Proposition 2 (Completeness and soundness). *Given an SSMT formula $\Phi = Pre : \varphi$ and the lower and upper probability thresholds t_l, t_u, we have:*

1. *The deduction relation $\longrightarrow_{SSMT}$ is terminating when iteratively applied to $(Pre, \varphi, \emptyset, t_l, t_u)$ as start of the deduction sequence. Each terminal state x has the form $x = (p, \varphi')$ with $p \in [0, 1]$ and φ' being an SMT formula.*

2. If $(Pre, \varphi, \emptyset, t_l, t_u) \longrightarrow^*_{SSMT} (p, \varphi')$ *then* $p > t_u$ *if* $Pr(\Phi) > t_u$*, and* $p < t_l$ *if* $Pr(\Phi) < t_l$*, and* $p = Pr(\Phi)$ *if* $t_l \leq Pr(\Phi) \leq t_u$*.*

Example of the SSMT algorithm. Consider target thresholds $t_l = 0.45$ and $t_u = 0.52$ and formula $\Phi = \exists x \in \{0,1\}\ \text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\} : (x > 0 \vee 2a + 4b \geq 3) \wedge (y > 0 \vee 2a + 4b < 1)$ from Fig. 1. The initial proof state is $(\exists x \in \{0,1\}\ \text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi, \emptyset, 0.45, 0.52)$, where $\varphi = (x > 0 \vee 2a + 4b \geq 3) \wedge (y > 0 \vee 2a + 4b < 1)$. Only rules (8), (10), or (13) are applicable, each involving a choice over the domain of the leading quantifier. To determine the rule to apply, we choose the value 0 for x and obtain $(\text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi, \langle x = 0\rangle, 0.45, 0.52)$. Then, (16) gives us the theory atom $2a + 4b \geq 3$, i.e. we go to $(\text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi, \langle x = 0, 2a + 4b \geq 3\rangle, 0.45, 0.52)$. Here, we select value 0 with probability 0.6 for the randomized variable y whence the maximum possible remaining probability is 0.4. This gives state $(\emptyset, \varphi, M = \langle x = 0, 2a+4b \geq 3, y = 0\rangle, t'_l, t'_u)$ where $t'_l = (0.45 - 0.4)/0.6$ and $t'_u = 0.52/0.6$. This triggers the deduction of $2a + 4b < 1$ again by rule (16). We thus encounter a conflict since the two theory atoms are inconsistent, and learn, e.g., the conflict clause $(x \neq 0 \vee y \neq 0)$. Hence, $(\emptyset, \varphi, M, t'_l, t'_u) \longrightarrow^*_{SSMT} (0, \varphi')$ where $\varphi' = \varphi \wedge (x \neq 0 \vee y \neq 0)$. By rule (12), this yields $(\text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi, \langle x = 0\rangle, 0.45, 0.52) \longrightarrow_{SSMT} (0, \varphi')$ since t_l can no longer be attained due to $0.45 - (0.6 \cdot 0) > 0.4$. Therefore, neither rule (10) nor rule (13) are applicable wrt. the initial situation. We thus try to establish the preconditions of the remaining rule (8), i.e. we investigate the other branch for x, continuing from proof state $(\exists x \in \{1\}\ \text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi', \langle x \neq 0\rangle, 0.45, 0.52)$. Selecting value 1 for x by rule (10) and value 0 for y by the choices opening rules (11) and (12) leads to a satisfying branch, i.e. $(\emptyset, \varphi', \langle x \neq 0, x = 1, y = 0, 2a+4b < 1\rangle, (0.45-0.4)/0.6, 0.52/0.6) \longrightarrow_{SSMT} (1, \varphi')$. Then, rule (11) matches and we obtain $(\text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi', \langle x \neq 0\rangle, 0.45, 0.52) \longrightarrow_{SSMT} (0.6, \varphi')$ since the computed satisfaction probability 0.6 exceeds $t_u = 0.52$. Finally, application of rule (8) yields $(\exists x \in \{0,1\}\ \text{ꓤ}_{\langle(0,0.6),(1,0.4)\rangle} y \in \{0,1\}, \varphi, \emptyset, 0.45, 0.52) \longrightarrow_{SSMT} (0.6, \varphi')$. Since the computed probability bound $p = 0.6$ is greater than $t_u = 0.52$, the maximum probability of satisfying Φ must exceed threshold 0.52. The thus computed value p is just a lower bound of $Pr(\Phi)$ (which is 1, cf. Fig. 1), but sufficient as a witness of probabilitistic satisfaction.

6 Conclusion and future work

This paper has given a detailed account of a fully symbolic encoding of probabilistic bounded reachability problems of discrete-time probabilistic hybrid automata, together with a generalized SMT procedure permitting the symbolic analysis of that encoding. Together, the two provide the germs of fully symbolic techniques for analyzing probabilistic hybrid systems without resorting to approximation by intermediate finite-state abstractions, thus potentially enhancing accuracy and scalability of the analysis algorithms. Implementation of these algorithms by means of an extension of the iSAT solver [12] with randomized quantifiers and the pertinent deduction rules has recently commenced, with a first prototype being operational, see `http://sisat.gforge.avacs.org/`.

References

1. L. Arnold. *Stochastic Differential Equations: Theory and Applications.* Wiley - Interscience, 1974.
2. G. Audemard, M. Bozzano, A. Cimatti, and R. Sebastiani. Verifying industrial hybrid systems with MathSAT. In *BMC*, *ENTCS* 119:17–32, 2004.
3. A. Biere, A. Cimatti, and Y. Zhu. Symbolic model checking without BDDs. In *TACAS*, *LNCS* 1579:193–207, 1999.
4. H. A. P. Blom, J. Krystul, and G. J. Bakker. A particle system for safety verification of free flight in air traffic. In *Decision and Control*, pages 1574–1579. IEEE, 2006.
5. L. Bordeaux and H. Samulowitz. On the stochastic constraint satisfaction framework. In *SAC*, pages 316–320. ACM, 2007.
6. L. Bujorianu and J. Lygeros. Toward a general theory of stochastic hybrid systems. In *Stochastic Hybrid Systems: Theory and Safety Critical Applications*, *LNCIS* 337:3–30, 2006.
7. M. L. Bujorianu and J. Lygeros. Reachability questions in piecewise deterministic Markov processes. In *HSCC*, *LNCS* 2623: 126–140, 2003.
8. M. Davis. *Markov Models and Optimization.* Chapman & Hall, London, 1993.
9. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Comm. of the ACM*, 5:394–397, 1962.
10. B. Dutertre and L. de Moura. A Fast Linear-Arithmetic Solver for DPLL(T). In *CAV*, *LNCS* 4144:81–94, 2006.
11. M. Fränzle and C. Herde. HySAT: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*, 30:179–198, 2007.
12. M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Efficient Solving of Large Non-linear Arithmetic Constraint Systems with Complex Boolean Structure. *Journal on Satisfiability, Boolean Modeling and Computation*, 1:209–236, 2007.
13. J. F. Groote, J. W. C. Koorn, and S. F. M. van Vlijmen. The Safety Guaranteeing System at Station Hoorn-Kersenboogerd. In *Conference on Computer Assurance*, pages 57–68. National Institute of Standards and Technology, 1995.
14. E. C. R. Hehner. Predicative programming. *Comm. of the ACM*, 27:134–151, 1984.
15. J. P. Hespanha. Polynomial stochastic hybrid systems. In *HSCC*, *LNCS* 3414:322–338, 2005.
16. J. Hu, J. Lygeros, and S. Sastry. Towards a theory of stochastic hybrid systems. In *HSCC*, *LNCS* 1790:160–173, 2000.
17. X. D. Koutsoukos and D. Riley. Computational methods for reachability analysis of stochastic hybrid systems. In *HSCC*, *LNCS* 3927:377–391, 2006.
18. M. L. Littman, S. M. Majercik, and T. Pitassi. Stochastic Boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001.
19. R. Nieuwenhuis, A. Oliveras, and C. Tinelli. Solving SAT and SAT Modulo Theories: from an Abstract Davis-Putnam-Logemann-Loveland Procedure to DPLL(T). *Journal of the ACM*, 53(6):937–977, 2006.
20. J. Sproston. *Model Checking of Probabilistic Timed and Hybrid Systems.* PhD thesis, University of Birmingham, 2000.