PSMT: Satisfiability Modulo Theories Meets Probability Distribution

Fuqi Jia [1,3] *, Rui Han [1,3] *, Xutong Ma [1,3], Baoquan Cui [1,3], Minghao Liu [1,3],
Pei Huang [4], Feifei Ma [2,3] †, and Jian Zhang [1,3] †
[1]SKLCS, ISCAS, Beijing, China
[2]SKLCS, LPSCS, ISCAS, Beijing, China
[3]University of Chinese Academy of Sciences, Beijing, China
[4]Stanford University, Stanford, USA
{jiafq,maff,zj}@ios.ac.cn

***Abstract*—SMT (Satisfiability Modulo Theories) has been widely used in program verification, analysis, and test generation. But sometimes, SMT solver outputs incomprehensible solutions, especially for practical instances. Besides, due to the design of the deterministic algorithms, for a given formula, the result of each run is the same. In this paper, we concentrate on combining SMT solving with probability, which will instruct the SMT solver to give some plausible solutions.**

We define a special problem: PSMT, which allows solving an SMT instance with variables conforming to a certain distribution. We define distribution under constraint for PSMT, which is based on MCSAT (Model Constructing Satisfiability), a mainstream SMT-solving algorithm. We propose the Prob-MCSAT algorithm, which combines the MCSAT algorithm and introduces the probability to variables. The visualized examples show that the resulting assignments will form a clear trend based on Prob-SMT.

***Index Terms*—Satisfiability Modulo Theories, Model Constructing Satisfiability, Probability Distribution.**

I. Introduction

SMT (Satisfiability Modulo Theories) solvers are powerful tools used in various domains for solving complex logical constraints involving combinations of different theories. It has many applications, including model checking [1], software verification [2], program analysis [3], test generation [4] etc.

Despite this, in daily use, there are still some minor problems. For example, the given solutions are not plausible and they will not be very good tests for real-world situations. It is related to the internal algorithm of the SMT solver. Most algorithms and implementations prefer boundary values, like the general simplex algorithm for linear theory [5].

```
void HealthCare(double weight, double height){
    BMI = weight / (height * height);
    if(BMI >= 18.5 && BMI <= 24.9) {...} // healthy
    else {...} // treatment suggestion
}
```

Motivation Example. The Body Mass Index (BMI) is used to assess the health status in demographic studies roughly, and the healthy range for adults is $[18.5, 24.9]$ [6]. Solving via an SMT solver in the healthy path (line 3) will result in an assignment $w = 0.375, h = 0.125$ where w and h denote a person's weight and height. Although this test case is correct, it seems very uncommon.

In many application scenarios, a variable usually follows a certain probability distribution. We can add a probability distribution, for example, $w \sim N(60, 10), h \sim N(1.7, 0.1)$, which means w follows the normal distribution with a mean value of 60 and standard deviation of 10, similarly for h of 1.7 and 0.1 respectively.

In the following text, we will introduce how to combine SMT with probability distributions and then propose a solving algorithm named Prob-MCSAT based on MCSAT (Model Constructing Satisfiability) algorithm [7]. After adding the probability distributions, Prob-MCSAT can obtain an assignment $w = 66.82, h = 1.75$, which seems a plausible solution.

In summary, our contributions and values are mainly:

1) We introduce the probability distribution into SMT and name it the PSMT problem. It allows the variables in solutions to satisfy certain probability distributions.
2) Based on the MCSAT algorithm, we define the *Distribution under Constraint* and construct the Prob-MCSAT algorithm, combining probability distribution and conflict-driven process. Prob-MCSAT obtains assignments that satisfy constraints and *Distribution under Constraint*.
3) Given distributions for variables, Prob-MCSAT can generate a plausible solution. It can also be used for sampling, where the resulting assignments will show a certain trend in the satisfiable space. Especially for some real-world scenarios, the produced solutions, for example as test cases, are closer to reality.

II. Background

A. MCSAT framework

MCSAT [7] is widely used to solve satisfiability problems involving propositional formulas with first-order logic. Compared with traditional SMT solvers based on the DPLL(T) framework [8], MCSAT has not a theory solver as a black box but organically integrates it into the solving process as part of making an assignment. For theories where the theory solver is relatively inefficient, for example, nonlinear int arithmetic [9], MCSAT always performs better than DPLL(T) [10].

* These authors contributed equally.
† Corresponding authors.

Under the MCSAT framework, the solver initializes an empty trail to store the partial assignment and inferred lemmas. According to the input formula, the solver uses the corresponding theory solver to make decisions and deductions on the constraints, discovering new lemmas or contradictions. The new information is used to update the current assignment, where the updating process involves operations such as checking, backtracking, and resolving. The process is repeated until an assignment that satisfies the constraints is found or it is determined that no solution exists.

B. Satisfiability Modulo Theories with Probability

There are some works combining SMT with probability, where SMT solvers offer additional capabilities for solving problems involving logical constraints and probabilistic reasoning. They have applications in many fields like probabilistic program analysis [11] and stochastic hybrid systems [12]. A technique known as SSMT (Stochastic Satisfiability modulo Theories) [13] has been proposed for depth-bounded reachability. All the aforementioned works concentrate on the probability that the constraints are satisfied. Recent work on SMT sampling [14] approximates sample points by adding extra constraints after the solver obtains a solution.

III. PSMT: Probability Satisfiability Modulo Theories

The PSMT problem asks for an assignment that follows the pre-defined "distribution" modified by constraints. A constraint denotes the logical formulas of an SMT problem. Given a constraint ψ, $\psi(\alpha)$ denotes the simplified constraint via substituting the partial assignment α into ψ. In order to define the probability distribution on constraints, we should define domains for variables first.

Definition 1 (Domain under Constraint)**.** *Given a constraint ψ and variable x, the domain under constraint $D(\psi, x)$ is the domain of x where when assigning x any value in $D(\psi, x)$, there exists a full assignment satisfying ψ.*

For complex constraints, obtaining the domain at the beginning is almost impossible. The interdependence of multiple variable assignments greatly hinders obtaining such domains. Due to the difficulty of obtaining the *Domain under Constraint*, defining the joint distribution function for the SMT problem at the beginning is also impractical. So we allow variables to pre-specify a value distribution in $\mathbb{R}$ and define the *Distribution under Constraint*.

Definition 2 (Distribution under Constraint)**.** *Given a constraint ψ and a distribution for variable x, whose probability density function is $P(x), x \in \mathbb{R}$, the distribution of x under constraint ψ is a refined distribution whose probability density function is $\widetilde{P}(\psi, x)$,*

$$\widetilde{P}(\psi, x) = \begin{cases} \dfrac{1}{\int_{D(\psi,x)} P(x)dx} P(x), x \in D(\psi, x), \\ 0, x \in \mathbb{R} - D(\psi, x). \end{cases}$$

The integral of $\widetilde{P}(\psi, x)$ over the domain is 1. The *Distribution under Constraint* gives the distribution that a variable follows, and based on it, we can derive the definition of PSMT.

Definition 3 (Probability Satisfiability Modulo Theories)**.** *Given an n variable SMT constraint ψ, a value distribution P, and a variable order σ, find an assignment $\alpha \models \psi$, i.e., α satisfies ψ. Meanwhile, the i-th variable x follows distribution under constraint, i.e., $\alpha[x_i] \sim \widetilde{P}(\psi(\{\alpha[x_{\sigma_1}], \cdots, \alpha[x_{\sigma_{i-1}}]\}), x_i)$ where $1 \leq i \leq n$.*

Example 1. *Given a constraint $\psi = \{-2 \leq x \leq 2 \wedge x^2 + y \geq 1\}$, and x follows a uniform distribution in $[-2, 2]$, i.e., $U([-2, 2])$. Note that the parameter of U is an interval, which means that the variable follows a uniform distribution in the interval. If a partial assignment is $\alpha = \{y \leftarrow 0\}$, then we have $\psi(\alpha) = -2 \leq x \leq 2 \wedge x^2 \geq 1$. The Domain under Constraint will be $D(\psi(\alpha), x) = [-2, -1] \cup [1, 2]$. We can derive the Distribution under Constraint with probability density function,*

$$\widetilde{P}(\psi(\alpha), x) = \frac{1}{\int_{-2}^{-1} \frac{1}{4} dx + \int_{1}^{2} \frac{1}{4} dx} P(x) = 2P(x) = \frac{1}{2},$$

where $x \in D(\psi(\alpha), x)$. In this example, we know $x \sim U([-2 - 1] \cup [1, 2])$. If we sample a value v for x under the distribution, a full assignment $\{x \leftarrow v, y \leftarrow 0\}$ is a solution to the PSMT problem.

Algorithm 1 Prob-MCSAT

Input: A set of real variables V, a constraint ψ.
Output: The status of the constraint (SAT/UNSAT).

$\mathcal{M} \leftarrow []$. // Initialize an empty trail.
while Exists an unassigned variable **do**
  // Assign
  $x \leftarrow$ the selected variable.
  $v \leftarrow$ GetAssignment(x, P).
  Propagate($\{x \leftarrow v\}$): Attempt to bring the assignment into ψ until no new assignment or conflict.
  if No Conflict **then**
    $\mathcal{M} \leftarrow \mathcal{M} \cup \{x \leftarrow v\}$.
  else
    $l \leftarrow$ the learned lemma to resolve the conflict.
    if l is $\perp$ **then**
      return UNSAT.
    else
      // Backtrack
      Remove assignment and lemmas from $\mathcal{M}$ until no conflict in ψ and add the learned lemma l to $\mathcal{M}$.
    end if
  end if
end while
return SAT.

IV. MCSAT WITH PROBABILITY

A. Prob-MCSAT Algorithm

Algorithm 1 is a modified version of MCSAT, especially GetAssignment in Line 5, stated in Algorithm 2. We assign a value according to *Distribution under Constraint* to the selected variable.

The Prob-MCSAT also contains three key components: *Assign*, *Propagate*, and *Backtrack*. *Assign* attempts to select a variable and assign it a value (Lines 3-5); *Propagate* brings the assignment to constraint to check whether the current partial assignment is satisfiable (Lines 6-17); when encountering conflict, *Backtrack* attempts to resolve the conflict via a theory lemma and then go back to a candidate satisfiable state (Lines 9-17); Note that if a conflict cannot be resolved, then we know that the constraint is unsatisfiable. When the union of unsatisfiable intervals of a variable is $\mathbb{R}$, there will be a conflict. It happens when propagating a new assignment to constraint and can be detected by some theory engines. For example, $x \leftarrow 0$ will result in a conflict in $y^2+1 = x$, because the unsatisfiable interval of y becomes $\mathbb{R}$.

In the conflict-driven process, unsatisfiable intervals for variables will gradually be built. It helps to build the *Distribution under Constraint* via building the *Domain under Constraint*. Note that at the beginning, the *Domain under Constraint* is not accurate and gradually becomes clear and precise as Prob-MCSAT runs.

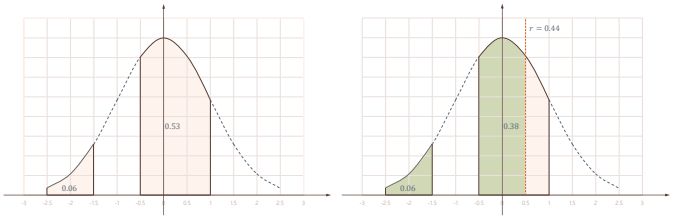
B. GetAssignment Algorithm

The unsatisfiable interval set is maintained globally and modified inside the Prob-MCSAT algorithm when propagating and backtracking. In Algorithm 2 (Lines 1-3), we convert the unsatisfiable interval set to the satisfiable interval set via complement in $\mathbb{R}$, i.e., $I' \leftarrow \mathbb{R} - \bigcup_{i=1}^{n} I[i]$, where I is the unsatisfiable interval set, n is the size of I, and I' is the satisfiable interval set.

If the union of the satisfiable interval set is $\mathbb{R}$, which means no constraint limits x, we can sample the value directly under the distribution (Lines 4-7). Note that each variable may indeed be unconstrained at the beginning, but as Prob-MCSAT runs, the unsatisfiable intervals of the variable will be replenished according to the conflict-driven process.

Given a distribution with probability density function $P(x)$, its cumulative distribution function $\mathit{CDF}(v) = \int_{-\infty}^{v} P(x)dx$ and the inverse cumulative distribution function $\mathit{ICDF}(t) = v$. As stated in [15], we can sample a value v under the uniform distribution and $\mathit{CDF}(v) = t$. According to *ICDF*, we obtain a sample point v under $P(x)$, i.e., $v = \mathit{ICDF}(t)$. But after the conflict-driven process, $\mathbb{R}$ will be split into some fragments. It is hard to compute the segmented *ICDF* on fragmented intervals. As an example in Fig. 1a, after splitting unsatisfiable intervals $(-\infty, -2.5) \cup (-1.5, -0.5) \cup (1, +\infty)$ from $\mathbb{R}$, the satisfiable intervals are $[-2.5, -1.5] \cup [-0.5, 1]$.

We obtain a sample point via Lines 8-27 in the GetAssignment algorithm. It first samples r under uniform distribution in $[0, s]$ where s is the sum of probabilities of fragmented



(a) Segmented Distribution (b) Sample in Segmented Distribution

Fig. 1: A normal distribution on fragmented satisfiable intervals ($[-2.5, -1.5] \cup [-0.5, 1]$) of a variable.

intervals of the original distribution. Note that since $\widetilde{P}(\psi, x)$ and $P(x)$ are proportional as in Definition 2, sampling from $\widetilde{P}(\psi, x)$ in $[0, 1]$ and sampling from $P(x)$ in $[0, s]$ is same. So we utilize $P(x)$ instead of computing $\widetilde{P}(\psi, x)$ for simplifying. We must restore r to the original distribution $P(x)$ as its *ICDF* is already known. The distribution of the final sample point

$$CDF(v) = CDF(L[k]) + r - \sum_{i=1}^{k-1} Q[i],$$

where $Q[i]$ is the cumulative distribution of i-th interval, k is the maximum index where $r - \sum_{i=1}^{k-1} Q[i] > 0$, and $L[k]$ is lower bound of k-th interval. So we obtain the sample point $v = \mathit{ICDF}(t) = \mathit{ICDF}(\mathit{CDF}(v))$. For example, we sample $r \leftarrow 0.44$ in $[0, 0.59]$, and $r - 0.06 > 0$ tells that we should sample in the second interval. The sample point is v and $\mathit{CDF}(v) = r - 0.06 + \mathit{CDF}(-0.5) \approx 0.68$. Finally, we can obtain $v = \mathit{ICDF}(\mathit{CDF}(v)) \approx 0.5$.

V. CASE STUDY

We implemented the tool [1] based on Z3 4.12.1. The tool now supports two types of distribution: U, uniform distribution; N, normal distribution. We concentrate on Example 2 [16]. and run each parameter setting 1000 times.

Example 2. *Assume we collect path constraints like*

$$x - y \geq 0 \wedge x + y \leq 0 \wedge y \geq -1.$$

We want to sample in such a constrained area.

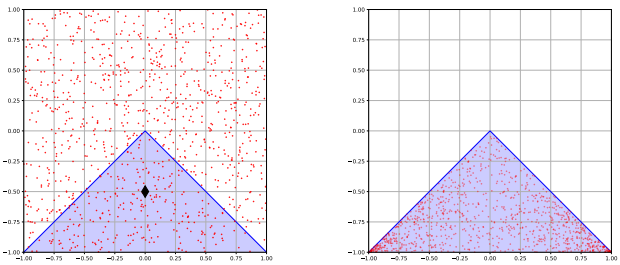
We set $x \sim U([-1, 1])$ and $y \sim U([-1, 1])$. A uniform sampler without a conflict-driven process will generate sample points like Fig. 2a. Prob-MCSAT with uniform distribution will generate sample points like Fig. 2b. The difference between them is that a uniform sampler will generate sample points without considering the satisfiability, and Prob-MCSAT will generate sample points in the satisfiable region following *Distribution under Constraint*. At the two bottom corners of the triangle, the sampling points of Fig. 2b will be obviously clustered, which means that $y \sim U([-1, 1])$ does not hold. When x approaches -1 or 1, the satisfiable interval of y will shrink, and y follows $U([-1, x])$ or $U([-1, -x])$, respectively, so the sample points in the region will appear very dense. The black diamond in Fig. 2a results from Z3, which is invariant within running 1000 times.

[1] https://github.com/Ailuras/prob-smt

Algorithm 2 GetAssignment

Input: A variable x and a distribution with corresponding probability density function $P(x)$.
Output: A satisfying assignment to x.

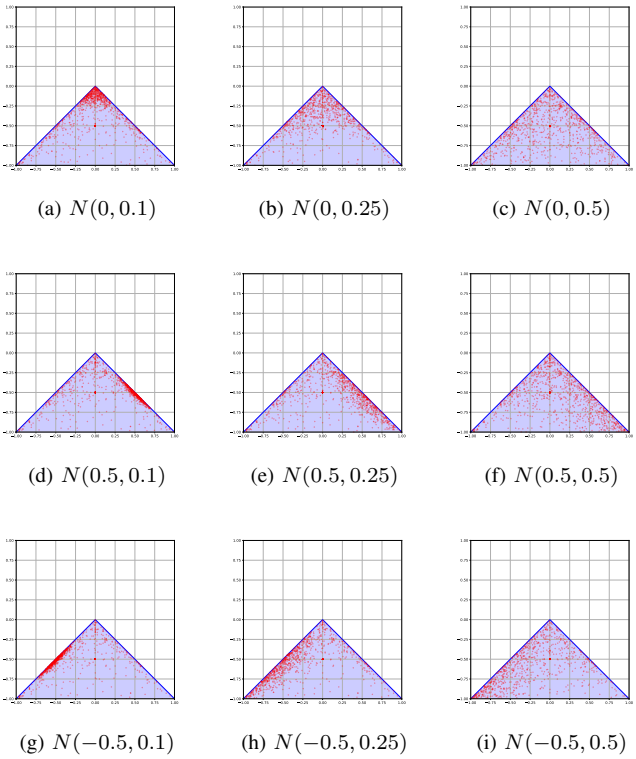
$I \leftarrow$ current unsatisfiable interval set of x.
$I \leftarrow$ Complement(I).
$n \leftarrow$ the size of I.
if $I = \mathbb{R}$ **then**
  $v \leftarrow$ sample under $P(x)$.
  return v.
end if
$s \leftarrow 0$.
$Q[i] \leftarrow 0, i = 1 \cdots n$.
for $i = 1 \cdots n$ **do**
  $L[i] \leftarrow$ lower bound of i-th satisfiable interval.
  $U[i] \leftarrow$ upper bound of i-th satisfiable interval.
  $Q[i] \leftarrow CDF(U[i]) - CDF(L[i])$.
  $s \leftarrow s + Q(i)$.
end for
$r \leftarrow$ a random number in $[0, s]$.
$k \leftarrow 0$.
while TRUE **do**
  if $r - Q(k) \leq 0$ **then**
    break.
  end if
  $r \leftarrow r - Q(k)$.
  $k \leftarrow k + 1$.
end while
$t \leftarrow CDF(L[k]) + r$.
$v \leftarrow ICDF(t)$.
return v.



(a) Uniform Sampler (b) Uniform Prob-MCSAT

Fig. 2: The difference between Uniform Sampler and Uniform Prob-MCSAT.

Besides, we also show the performance of normal distribution in Fig. 3, where $x \sim N(M, V)$ as stated in the captions and $y \sim N(0, 0.1)$. The sample points form an obvious tendency in the satisfiable space. Prob-MCSAT will first assign x and then y. When comparing the three figures in each row to each other (Fig. 3a - Fig. 3c, Fig. 3d - Fig. 3f, Fig. 3g - Fig. 3i), the center of the cluster of sample points does not change but presents a divergent state, which is caused by the



(a) $N(0, 0.1)$ (b) $N(0, 0.25)$ (c) $N(0, 0.5)$

(d) $N(0.5, 0.1)$ (e) $N(0.5, 0.25)$ (f) $N(0.5, 0.5)$

(g) $N(-0.5, 0.1)$ (h) $N(-0.5, 0.25)$ (i) $N(-0.5, 0.5)$

Fig. 3: Normal distribution Prob-MCSAT with different parameters. The caption is the distribution that x follows.

gradual increase of the variance. When comparing the three figures in each column to each other (Fig. 3a - Fig. 3g, Fig. 3b - Fig. 3h, Fig. 3c - Fig. 3i), the closeness of the cluster remains nearly the same, but the center has shifted, which is caused by the change of the mean.

As mentioned above, assigning probabilities to variables can help us to conduct different forms of sample tendencies in the satisfiable space to a certain extent.

VI. Conclusion and Future Work

In this paper, we propose a special problem: PSMT, which allows solving an SMT instance with variables conforming to certain distributions. Based on MCSAT, we define Distribution under Constraint and propose the Prob-MCSAT algorithm, which combines the MCSAT algorithm with introducing the probability of variables. The visualized examples show that Prob-MCSAT can produce a clear trend in the results. This method can also be explored further for test generation, especially in scenarios with real-world backgrounds as they usually have certain distribution characteristics.

Acknowledgement

This work has been supported by the National Natural Science Foundation of China (NSFC) under grants No.61972384 and No. 62132020. Feifei Ma is also supported by the Youth Innovation Promotion Association CAS under grant No. Y202034. The authors would like to thank the anonymous reviewers for their comments and suggestions.

REFERENCES

[1] L. C. Cordeiro, B. Fischer, and J. Marques-Silva, "SMT-Based bounded model checking for embedded ANSI-C software," in *ASE 2009, 24th IEEE/ACM International Conference on Automated Software Engineering, Auckland, New Zealand, November 16-20, 2009*. IEEE Computer Society, 2009, pp. 137–148.

[2] D. Beyer, M. Dangl, and P. Wendler, "A unifying view on SMT-Based software verification," *J. Autom. Reason.*, vol. 60, no. 3, pp. 299–335, 2018.

[3] N. Gavrilenko, H. P. de León, F. Furbach, K. Heljanko, and R. Meyer, "BMC for weak memory models: Relation analysis for compact SMT encodings," in *Computer Aided Verification - 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I*, ser. Lecture Notes in Computer Science, I. Dillig and S. Tasiran, Eds., vol. 11561. Springer, 2019, pp. 355–365.

[4] J. Peleska, E. Vorobev, and F. Lapschies, "Automated test case generation with SMT-Solving and abstract interpretation," in *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, ser. Lecture Notes in Computer Science, M. G. Bobaru, K. Havelund, G. J. Holzmann, and R. Joshi, Eds., vol. 6617. Springer, 2011, pp. 298–312.

[5] B. Dutertre and L. M. de Moura, "A fast linear-arithmetic solver for DPLL(T)," in *Computer Aided Verification, 18th International Conference, CAV 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, ser. Lecture Notes in Computer Science, T. Ball and R. B. Jones, Eds., vol. 4144. Springer, 2006, pp. 81–94.

[6] "Assessing your weight," https://www.cdc.gov/healthyweight/assessing/index.html, online.

[7] D. Jovanovic and L. M. de Moura, "Solving non-linear arithmetic," in *IJCAR 2012*, B. Gramlich, D. Miller, and U. Sattler, Eds., vol. 7364, 2012, pp. 339–354.

[8] 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 (JACM)*, vol. 53, no. 6, pp. 937–977, 2006.

[9] F. Jia, R. Han, P. Huang, M. Liu, F. Ma, and J. Zhang, "Improving bit-blasting for nonlinear integer constraints," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2023, Seattle, WA, USA, July 17-21, 2023*, R. Just and G. Fraser, Eds. ACM, 2023, pp. 14–25.

[10] D. Jovanovic, "Solving nonlinear integer arithmetic with MCSAT," in *VMCAI 2017*, A. Bouajjani and D. Monniaux, Eds., vol. 10145, 2017, pp. 330–346.

[11] M. Borges, A. Filieri, M. d'Amorim, and C. S. Păsăreanu, "Iterative distribution-aware sampling for probabilistic symbolic execution," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, 2015, pp. 866–877.

[12] A. Abate, J.-P. Katoen, J. Lygeros, and M. Prandini, "Approximate model checking of stochastic hybrid systems," *European Journal of Control*, vol. 16, no. 6, pp. 624–641, 2010.

[13] M. Fränzle, H. Hermanns, and T. Teige, "Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems," in *Hybrid Systems: Computation and Control: 11th International Workshop, HSCC 2008, St. Louis, MO, USA, April 22-24, 2008. Proceedings 11*. Springer, 2008, pp. 172–186.

[14] M. Peled, B. Rothenberg, and S. Itzhaky, "SMT sampling via model-guided approximation," in *Formal Methods - 25th International Symposium, FM 2023, Lübeck, Germany, March 6-10, 2023, Proceedings*, ser. Lecture Notes in Computer Science, M. Chechik, J. Katoen, and M. Leucker, Eds., vol. 14000. Springer, 2023, pp. 74–91.

[15] M. Borges, A. Filieri, M. d'Amorim, and C. S. Pasareanu, "Iterative distribution-aware sampling for probabilistic symbolic execution," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering, ESEC/FSE 2015, Bergamo, Italy, August 30 - September 4, 2015*, E. D. Nitto, M. Harman, and P. Heymans, Eds. ACM, 2015, pp. 866–877.

[16] M. B. Dwyer, A. Filieri, J. Geldenhuys, M. J. Gerrard, C. S. Pasareanu, and W. Visser, "Probabilistic program analysis," in *Grand Timely Topics in Software Engineering - International Summer School GTTSE 2015, Braga, Portugal, August 23-29, 2015, Tutorial Lectures*, ser. Lecture Notes in Computer Science, J. Cunha, J. P. Fernandes, R. Lämmel, J. Saraiva, and V. Zaytsev, Eds., vol. 10223. Springer, 2015, pp. 1–25.