Exact Symbolic Reasoning for Nonlinear Stochastic SMT via Cylindrical Algebraic Decomposition

Jung-Cheng Lin
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

Chia-Hsuan Su
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

Jie-Hong R. Jiang
Department of Electrical Engineering, National Taiwan University, Taipei, Taiwan
Graduate Institute of Electronics Engineering, National Taiwan University, Taipei, Taiwan

Hiroshi Unno
Research Institute of Electrical Communication, Tohoku University, Sendai, Japan

Abstract

Stochastic Satisfiability Modulo Theories (SSMT) has traditionally focused on the interplay between existential and randomized quantifiers, typically relying on numerical sampling or approximations. We present a generalized SSMT framework that integrates universal quantification, lifting the formalism to a robust stochastic game-theoretic setting. By treating universal quantifiers as the adversarial infimum of satisfaction probabilities, our framework enables the exact modeling of competitive interactions under uncertainty.

Our approach leverages Cylindrical Algebraic Decomposition (CAD) to derive exact symbolic probability expressions for Nonlinear Real Arithmetic (NRA) formulas, moving beyond the limitations of linear constraints and point-value estimations. Central to our contribution is a recursive quantifier elimination algorithm designed to handle variable-dependent domains and non-algebraic expressions through a variable reparameterization technique.

Experimental evaluation across baseline synthetic formulas, strategic economic models, and probabilistic program verification benchmarks demonstrates that our framework consistently computes exact symbolic solutions. By achieving a degree of symbolic precision and expressiveness unattainable by traditional numerical solvers, this work establishes a new baseline for exact reasoning in stochastic adversarial environments.

2012 ACM Subject Classification Theory of computation → Automated reasoning

Keywords and phrases Stochastic Satisfiability Modulo Theories (SSMT), Cylindrical Algebraic Decomposition (CAD), Quantifier Elimination

Digital Object Identifier 10.4230/LIPIcs.SAT.2026.24

Supplementary Material *Software (Repository)*: `https://github.com/NTU-ALComLab/SSMT`
archived at `swh:1:dir:a0162a93f4e0e485db04c0f92b9a7b93ff2e7ba0`

Funding This work was supported in part by the National Science and Technology Council of Taiwan under Grant NSTC 114-2221-E-002-183-MY3, by National Taiwan University under Grant NTU-DE114FR032, and by JSPS KAKENHI of Japan under Grant JP25K24739.

1 Introduction

The exploration of quantified formulas in Satisfiability Modulo Theories (SMT) has historically followed two divergent paths: Quantified SMT (QSMT) and Stochastic SMT (SSMT). QSMT research, exemplified by QSAT [3], QSMA [4], and fine-grained SimSat [19], centers on the deterministic interplay between universal (∀) and existential (∃) quantifiers, typically within Linear Real Arithmetic (LRA) and Linear Integer Arithmetic (LIA).

In contrast, SSMT was introduced by Fränzle et al. [8] to handle uncertainty by incorporating randomized quantifiers ($\mathfrak{R}$), effectively extending Stochastic SAT (SSAT) [14, 20] to hybrid domains. While subsequent research extended this to non-linear arithmetic [23] and continuous probability distributions [9, 10], these traditional approaches are largely restricted to formulas with only $\mathfrak{R}$ and $\exists$ quantifiers and rely on interval-based numerical approximations or sampling to estimate satisfaction probabilities. They targeted applications such as probabilistic hybrid automata [8], networked automation systems [9], and safety-critical verification [8, 10].

In this work, we introduce a comprehensive SSMT framework that, for the first time, integrates universal quantifiers ($\forall$) into the stochastic setting. By leveraging Cylindrical Algebraic Decomposition (CAD), our approach transcends the linear constraints and approximate reasoning of previous research, enabling the derivation of exact symbolic probability expressions for formulas in Non-linear Real Arithmetic (NRA).

While traditional SSMT models planning under uncertainty via the interplay of existential ($\exists$) and randomized ($\mathfrak{R}$) quantifiers, the inclusion of the universal quantifier ($\forall$) is essential for capturing adversarial environments. Mathematically, $\forall$ represents the infimum of satisfaction probabilities, establishing a rigorous lower bound on performance under the most unfavorable conditions. This addition elevates SSMT from a tool for probabilistic reachability to a stochastic game framework, providing a more expressive language for systems where strategic decisions, both existential and universal, must be made amidst inherent probabilistic uncertainty. Such expressiveness is vital for high-stakes domains, such as competitive economic modeling or analysis of security or safety-critical applications, where a system must guarantee a high success probability against strategic opponents or worst-case perturbations [1, 8].

The rest of this paper is organized as follows. After the preliminaries of Cylindrical Algebraic Decomposition are introduced in Section 2, Section 3 provides the syntax and recursive semantics for our generalized SSMT formulas. Section 4 details the main algorithm. In Section 5, we present extensions for the encoding of Boolean variables and a reparameterization technique for handling specific non-semi-algebraic constraints. The experimental results are shown in Section 6. Finally, Section 7 concludes this work.

2 Preliminaries

We represent the Boolean constants TRUE and FALSE by $\top$ and $\bot$. The notation $x \models \varphi$ indicates that the assignment x satisfies the formula φ. The Boolean operators $\neg$, $\wedge$, $\vee$, $\oplus$, and $\leftrightarrow$ are understood according to their standard logical interpretations. In addition, we employ the if–then–else operator, defined by $\text{ite}(x, y, z) \coloneqq (x \wedge y) \vee (\neg x \wedge z)$.

2.1 Real Closed Fields and Semi-algebraic Sets

A *real closed field* is an ordered field that shares the same first-order logical properties as the field of real numbers $\mathbb{R}$. In the context of SMT, this structure is fundamental because the theory of Real Closed Fields is decidable and admits quantifier elimination [5, 6].

This mathematical structure forms the basis for Nonlinear Real Arithmetic (NRA), the first-order theory that deals with polynomial constraints over the reals. The fundamental building blocks of NRA are atomic polynomial predicates (also called literals) of the form $f(x_1, \ldots, x_n) \sim 0$, where f is a real polynomial and $\sim \in \{=, >, \geq\}$. An NRA formula is defined as a finite Boolean combination of these literals using operators such as negation ($\neg$), conjunction ($\wedge$), and disjunction ($\vee$).

In real algebraic geometry, an *algebraic set* is a subset of $\mathbb{R}^n$ defined as the common zero locus of a finite collection of real polynomials, while a *semi-algebraic set* is a subset of $\mathbb{R}^n$ obtained from the solution sets defined by polynomial equalities and inequalities using finitely many set-theoretic operations such as union, intersection, and complement. Equivalently, semi-algebraic sets are defined by finite Boolean combinations of polynomial equalities and inequalities [2, 22]. A function $f : S \to \mathbb{R}^m$, where $S \subseteq \mathbb{R}^n$, is called *semi-algebraic* if the set $\{(x, f(x)) \mid x \in S\}$ is a semi-algebraic subset of $\mathbb{R}^{n+m}$ [22]. In particular, the solution set of any quantifier-free NRA formula is semi-algebraic. We refer to such formulas as *semi-algebraic expressions* throughout this paper. This property is essential in our framework, as it ensures that such satisfying regions remain semi-algebraic throughout the elimination process.

2.2 Cylindrical Algebraic Decomposition and Satisfying Cells

A *decomposition* partitions the n-dimensional real space $\mathbb{R}^n$, whose coordinates correspond to variables $x_1, \ldots, x_n$, into a finite collection of disjoint, connected, non-empty subsets called *cells*. A decomposition is *cylindrical* with respect to a given variable ordering $x_1 \prec x_2 \prec \cdots \prec x_n$ if, for every $i < n$, the projection of any cell onto the subspace spanned by the first i variables $(x_1, \ldots, x_i)$ is either identical to or disjoint from the projection of any other cell. Intuitively, this means that the cells are arranged in a layered way along the variables: once the lower-dimensional structure over $(x_1, \ldots, x_i)$ is fixed, the cells extend "above" it along the higher-order variables, forming a recursive cylindrical structure.

▶ **Example 1.** Consider the cell

$$c_1 = \{(x, y) \mid 0 < x < 1, -\sqrt{1 - x^2} < y < \sqrt{1 - x^2}\}.$$

Let $x \prec y$. Projecting c_1 onto the x-axis yields the interval $(0, 1)$. For each x in this interval, the corresponding set of y values forms a vertical segment between $y = -\sqrt{1 - x^2}$ and $y = \sqrt{1 - x^2}$. This illustrates the *cylindrical* structure: the cell is obtained by taking a base interval on the x-axis and extending it along the y-direction according to boundary functions of x. In this sense, the cell "stacks" above the x-interval, forming a structure analogous to a vertical cylinder.

Cylindrical Algebraic Decomposition (CAD) [6] is a method that takes as input a finite set of polynomials in n variables and partitions $\mathbb{R}^n$ into a finite set of connected semi-algebraic *cells*. Within each cell, every input polynomial in the input set has a constant sign (positive, negative, or zero). This cylindrical structure makes CAD particularly useful for tasks such as quantifier elimination [5] and solving polynomial inequalities.

▶ **Definition 2** (CAD). *Let $P = \{f_1, \ldots, f_k\}$ be a finite set of polynomials in $\mathbb{R}[x_1, \ldots, x_n]$ and let $\mathcal{O}$ be a variable ordering $x_1 \prec x_2 \prec \cdots \prec x_n$. A **Cylindrical Algebraic Decomposition** of $\mathbb{R}^n$ with respect to P and $\mathcal{O}$, denoted as $\mathrm{CAD}(P, \mathcal{O})$, is a finite partition of $\mathbb{R}^n$ into connected semi-algebraic cells $\{c_1, \ldots, c_m\}$ such that:*

1. ***Sign-Invariance:** Each polynomial $f_j \in P$ is sign-invariant (positive, negative, or zero) on every cell c_i.*
2. ***Cylindricality:** The decomposition is cylindrical with respect to $\mathcal{O}$; that is, for every $i < n$, the projection of any two cells c_a, c_b onto the first i coordinates is either identical or disjoint.*

While $\mathrm{CAD}(P, \mathcal{O})$ operates on a set of polynomials to create a geometric partition, our algorithm requires identifying the specific regions where a logical formula φ holds. We introduce a formula-induced CAD as follows.

▶ **Definition 3** (Formula-Induced CAD). *Given a quantifier-free NRA formula φ and a variable ordering $\mathcal{O}$, let $P_\varphi = \{f_1, \ldots, f_k\}$ be the set of polynomials appearing in φ. We refer to the subset of CAD induced by φ as:*

$$\mathrm{CAD}_\varphi(\mathcal{O}) = \{c \in \mathrm{CAD}(P_\varphi, \mathcal{O}) \mid \exists e \in c,\ e \models \varphi\}. \tag{1}$$

▶ Remark 4. Since CAD ensures that the truth value of φ is invariant over each cell (Definition 2), CAD_φ effectively filters the geometric partition to retain only those cells where φ is TRUE. Membership in CAD_φ can be determined by evaluating φ at a single sample point from each cell.

▶ **Example 5.** Consider the formula $\varphi = (x^2{+}y^2 < 1){\wedge}(x > 0)$ in $\mathbb{R}^2$ with $P_\varphi = \{x^2{+}y^2{-}1, x\}$ and $\mathcal{O} = x \prec y$. It can be checked that $\mathrm{CAD}(P_\varphi, \mathcal{O})$ partitions $\mathbb{R}^2$ into 23 cells. Among them, some examples include:

- Let $c_1 = \{(x, y) \mid 0 < x < 1, -\sqrt{1-x^2} < y < \sqrt{1-x^2}\}$. Since any sample point in c_1 satisfies both $x^2 + y^2 < 1$ and $x > 0$, $c_1 \in \mathrm{CAD}_\varphi(\mathcal{O})$.
- Let $c_2 = \{(x, y) \mid -1 < x < 0, -\sqrt{1-x^2} < y < \sqrt{1-x^2}\}$. Although c_2 satisfies $x^2 + y^2 < 1$, it fails $x > 0$; thus $c_2 \notin \mathrm{CAD}_\varphi(\mathcal{O})$.
- Let $c_3 = \{(x, y) \mid x > 1, y \in \mathbb{R}\}$. This cell fails $x^2 + y^2 < 1$; thus $c_3 \notin \mathrm{CAD}_\varphi(\mathcal{O})$.

Note that CAD_φ contains only one cell, namely c_1, in contrast to the 23 cells of $\mathrm{CAD}(P_\varphi, \mathcal{O})$. In fact, using CAD_φ, instead of $\mathrm{CAD}(P_\varphi, \mathcal{O})$, can significantly reduce the number of cells considered in our quantifier elimination procedure.

3 SSMT Syntax and Semantics

3.1 Syntax

An SSMT formula Φ is defined in *prenex form* as $\Phi = Q : \varphi$, where Q is a quantifier prefix and φ is a quantifier-free matrix. The prefix consists of a sequence of quantifiers $Q = Q_1 x_1 \in \mathrm{dom}(x_1), \ldots, Q_n x_n \in \mathrm{dom}(x_n)$, where each $Q_i \in \{\exists, \forall, \mathfrak{R}_{\pi_i}\}$. Here, $\exists$ and $\forall$ denote existential and universal quantifiers, while $\mathfrak{R}_{\pi_i}$ represents a randomized quantifier associated with an integrable *probability density function* (PDF) π_i such that

$$\int_{\mathrm{dom}(x_i)} \pi_i(x_i)\, dx_i = 1.$$

The matrix φ is an SMT formula over *quantifier-free nonlinear real arithmetic* (NRA), described as a finite Boolean combination of atomic polynomial predicates of the form $f(x_1, \ldots, x_n) \sim 0$, where $\sim \in \{=, >, \geq\}$.

A variable in Φ is said to be *bound* if it lies within the scope of a quantifier in the prefix Q, and *free* otherwise. An SSMT formula is called *closed* if it contains no free variables, and *open* if it contains free variables. In the latter case, the free variables are regarded as externally fixed parameters. Unless otherwise stated, we assume an SSMT formula is closed, that is, every variable occurring in the matrix φ is quantified in the prefix Q.

The domain $\mathrm{dom}(x_i)$ for each variable is a real interval specified by semi-algebraic constraints, meaning that $\mathrm{dom}(x_i)$ is described by a set of polynomial equalities and inequalities whose feasible set is semi-algebraic. For each i, let $D_i(x_1, \ldots, x_i)$ denote the quantifier-free NRA formula characterizing $\mathrm{dom}(x_i)$. Notably, these domains may be *variable-dependent*, meaning the interval for x_i can be restricted by the values of previously quantified variables $x_1, \ldots, x_{i-1}$. For example, a formula might specify $y \in [0, x]$. Such dependencies are permitted because Cylindrical Algebraic Decomposition (CAD) is capable of processing general semi-algebraic relations among quantified variables.

For randomized quantifiers, for simplicity, the notation $\mathfrak{U}x \sim \mathcal{U}(a,b)$ is used to denote a uniform distribution when $\mathfrak{U}_\pi x \in [a,b]$ and $\pi(x) = \frac{1}{b-a}$. We restrict our attention to probability density functions for which integration over CAD-induced cells can be carried out symbolically within the semi-algebraic framework. In particular, this includes the polynomial density functions considered in our implementation and experimental benchmarks. This restriction is necessary because eliminating a randomized quantifier introduces integral terms, and the resulting probability expressions must remain representable as finite case distinctions compatible with CAD-based quantifier elimination.

Lastly, as the proposed approach relies on CAD, the framework is restricted to real-valued variables within real closed fields. Consequently, other SMT theories such as integers, bit-vectors, arrays, or uninterpreted functions are currently unsupported. However, Boolean structures are supported at the propositional level and can be encoded as polynomial constraints over the set $\{0,1\}$ when necessary (Section 5.1).

3.2 Semantics

We define the semantics of an SSMT formula $\Phi = Q : \varphi$ as its probability of satisfaction $\Pr(\Phi) \in [0,1]$.

We assume Φ is in *prenex form*, i.e., all quantifiers appear at the front and the matrix φ is quantifier-free. All quantified variables have unique names. Each quantifier domain $\mathrm{dom}(x_i)$ may depend on previously quantified variables and is a real interval.

▶ **Definition 6.** *Let $\Phi = Q : \varphi$ with $Q = Q_1 x_1 \in \mathrm{dom}(x_1), \ldots, Q_n x_n \in \mathrm{dom}(x_n)$. The semantics of Φ is defined recursively as follows.*

$$\textit{Case } \Phi' = \varphi\textit{:} \qquad \Pr(\Phi') = \begin{cases} 1 & \textit{if } \varphi \textit{ is satisfiable,} \\ 0 & \textit{otherwise.} \end{cases} \tag{2}$$

$$\textit{Case } \Phi' = \exists x_i : \Psi\textit{:} \qquad \Pr(\Phi') = \sup_{v \in \mathrm{dom}(x_i)} \Pr(\Psi[v/x_i]), \tag{3}$$

$$\textit{Case } \Phi' = \forall x_i : \Psi\textit{:} \qquad \Pr(\Phi') = \inf_{v \in \mathrm{dom}(x_i)} \Pr(\Psi[v/x_i]), \tag{4}$$

$$\textit{Case } \Phi' = \mathfrak{U}_{\pi_i} x_i : \Psi\textit{:} \qquad \Pr(\Phi') = \int_{\mathrm{dom}(x_i)} \Pr(\Psi[v/x_i])\, \pi_i(v)\, dv. \tag{5}$$

The formula Ψ above is an open SSMT formula with free variable x_i, and $\Psi[v/x_i]$ denotes the formula obtained by substituting every appearance of variable x_i by v in Ψ.

The randomized-quantifier case is restricted to instances for which the integral expressions introduced by quantifier elimination remain symbolically representable within the semi-algebraic framework. Also, allowing domains to depend on previously quantified variables is essential to enrich our SSMT's semantics.

▶ **Example 7.** Consider the two formulas:

$$\Phi_1 \;=\; \mathfrak{U}x \sim \mathcal{U}(0,1),\, \forall y \in [0,x].\, (x \geq y), \text{and}$$
$$\Phi_2 \;=\; \mathfrak{U}x \sim \mathcal{U}(0,1),\, \forall y \in [0,1].\, (x \geq y).$$

These formulas illustrate that variable-dependent domains can lead to different semantic values. In the first formula, the domain of y depends on x, restricting y to values not exceeding x. Hence, for every $x \in [0,1]$, the inner universal condition holds, and therefore $\Pr(\Phi_1) = 1$.

In contrast, in the second formula, the domain of y is independent of x. For any fixed $x < 1$, there exists $y \in [0,1]$ such that $y > x$, so the universally quantified formula only holds for $x = 1$, and therefore $\Pr(\Phi_2) = 0$. Consequently, the two formulas receive different semantic interpretations.

3.3 Game-Theoretic Interpretation

The recursive semantics of the satisfaction probability defined in Section 3.2 can be formally interpreted as a zero-sum stochastic game played between two competing players. This interpretation is rooted in game-theoretical semantics (GTS), which identifies the truth of a quantified formula with the existence of a winning strategy for a specific player [11].

In the SSMT framework, the game is played between an existential player and a universal player. In this setting, the existential player controls the variables quantified by $\exists$ and attempts to pick values that maximize the satisfaction probability $\Pr(\Phi)$. In contrast, the universal player, acting as an adversarial spoiler, controls the variables quantified by $\forall$ and seeks values that minimize this probability. The randomized quantifiers (Ɐ) represent moves by a non-strategic chance player, where values are drawn according to the specified PDF π. This framework extends traditional QSMT games [3, 19] by introducing probabilistic uncertainty, where the value of the game corresponds to the exact symbolic probability derived by CAD.

4 Algorithm

Our approach computes the probability of satisfaction $\Pr(\Phi)$ by recursively eliminating quantifiers from the SSMT formula. Given a formula $\Phi = Q_1 x_1 \dots Q_n x_n : \varphi$, we process the quantifier prefix from the innermost variable x_n to the outermost variable x_1. This elimination follows the variable ordering $\mathcal{O} = x_1 \prec x_2 \prec \cdots \prec x_n$ derived from the quantifier prefix.

Given a semi-algebraic cell c_i, let ϕ_{c_i} denote a quantifier-free NRA formula such that $\mathbf{x} \in c_i$ if and only if $\mathbf{x} \models \phi_{c_i}$. During quantifier elimination, formulas are transformed into *probability expressions*, which are finite case distinctions of the form

$$F = \sum_i [\![\phi_{c_i}]\!] \cdot f_i, \tag{6}$$

where ϕ_{c_i} is the NRA formula of cell c_i, $[\![\cdot]\!]$ denotes the Iverson bracket [13], and each f_i is the probability function associated with that cell. In the sequel, we refer to each term $[\![\phi_{c_i}]\!] \cdot f_i$ of the above probability expression as a *branch*, denoted as a pair (c_i, f_i), and refer to $[\![\phi_{c_i}]\!]$ as the *guard* of the branch. By the equation, quantifier elimination is applied to each branch independently, and the results are combined by a summation.

Algorithm 1 outlines the computation procedure, which is detailed in the following sections.

4.1 General Recursive Framework

We define the kth-step intermediate probability expression during quantifier elimination as

$$F_k(x_1, \dots, x_k) = \Pr(Q_{k+1} x_{k+1} \dots Q_n x_n : \varphi[x_1, \dots, x_k]), \tag{7}$$

where the variables in the brackets indicate the free variables that determine the domain of the probability expression. The quantifier elimination for formula $\Phi = Q : \varphi$ proceeds from the initial case by using the formula-induced CAD with respect to the matrix φ and variable ordering $\mathcal{O}$ following the prefix:

$$F_n(x_1, \dots, x_n) = \sum_{c \in \mathrm{CAD}_\varphi(\mathcal{O})} [\![\phi_c]\!] \cdot 1. \tag{8}$$

The expressions $F_{n-1}, \dots, F_0$ are then computed recursively, with the final value $F_0 = \Pr(\Phi)$.

Algorithm 1 Exact SSMT via CAD.

Input: A closed SSMT formula $\Phi = Q_1 x_1\, Q_2 x_2 \cdots Q_n x_n : \varphi$, the variable ordering $\mathcal{O} = x_1 \prec x_2 \prec \cdots \prec x_n$, and the induced probabilistic expression $F_n = \sum_{c \in \mathrm{CAD}_\varphi(\mathcal{O})} [\![\phi_c]\!] \cdot 1$.

Output: The exact satisfaction probability $F_0 = \Pr(\Phi)$.

$k \leftarrow n$
while $k > 0$ **do**
  $F' \leftarrow 0$, $Z \leftarrow \emptyset$
  if $Q_k = \mathcal{U}$ **then**
    for *every branch* $[\![\phi_i]\!] \cdot f_i$ *in* F_k **do**
      // let ϕ_i be $\phi_i'(x_1, \ldots, x_{k-1}) \wedge \ell_i(x_1, \ldots, x_{k-1}) \le x_k \le u_i(x_1, \ldots, x_{k-1})$
      $F' \leftarrow F' + [\![\phi_i']\!] \cdot \int_{\ell_i(x_1,\ldots,x_{k-1})}^{u_i(x_1,\ldots,x_{k-1})} f_i(x_1, \ldots, x_k)\, \pi_k(x_k)\, dx_k$
  else
    if $Q_k = \forall$ **then**
      $Z \leftarrow \mathrm{ProjectUncoveredRegion}(F_k, x_k)$ // find uncovered x_k regions
    $\mathcal{O}_\theta \leftarrow x_1 \prec \cdots \prec x_{k-1} \prec \theta \prec x_k$
    for *every branch* $[\![\phi_i]\!] \cdot f_i$ *in* F_k **do**
      $C_i \leftarrow \mathrm{CAD}_{\phi_i \wedge \theta = f_i}(\mathcal{O}_\theta)$
      for *every cell* $c_{ij} \in C_i$ **do**
        // extract the projected guard ϕ_{ij}' and the θ-bounds ℓ_{ij}, u_{ij} from c_{ij}
        if $Q_k = \exists$ **then**
          $F' \leftarrow F' + [\![\phi_{ij}']\!] \cdot u_{ij}(x_1, \ldots, x_{k-1})$
        if $Q_k = \forall$ **then**
          $F' \leftarrow F' + [\![\phi_{ij}']\!] \cdot \ell_{ij}(x_1, \ldots, x_{k-1})$
  $F_{k-1} \leftarrow \mathrm{MergeBranchesByGuard}(F', Q_k, Z)$
    // merge branches with the same guards
  $\mathcal{O} \leftarrow \mathcal{O} \setminus \{x_k\}$
  $k \leftarrow k - 1$
return F_0

The calculation of F_{k-1} from F_k depends on the quantifier associated with x_k:

$$\text{Case } \exists x_k: \quad F_{k-1}(x_1, \ldots, x_{k-1}) = \sup_{x_k} F_k(x_1, \ldots, x_k) \tag{9}$$

$$\text{Case } \forall x_k: \quad F_{k-1}(x_1, \ldots, x_{k-1}) = \inf_{x_k} F_k(x_1, \ldots, x_k) \tag{10}$$

$$\text{Case } \mathcal{U}_{\pi_k} x_k: \quad F_{k-1}(x_1, \ldots, x_{k-1}) = \int_{\mathrm{dom}(x_k)} F_k(x_1, \ldots, x_k) \cdot \pi_k(x_k)\, dx_k. \tag{11}$$

The elimination of any quantifier Q_k is performed by processing each branch (c_i, f_i) of the probability expression F_k independently. After processing, the resulting probability expression for F_{k-1} is rebuilt by aggregating the contributions of each branch. While the overarching recursive structure remains the same, the specific CAD constructions vary depending on the quantifier type, as detailed in the following sections.

▶ **Example 8.** Consider the SSMT formula

$$\exists x \in [0, \infty)\, \forall y \in [0, 10]\, \mathcal{U} z \sim \mathcal{U}(0, 20).\ \mathrm{ite}(y \ge x, x + y \ge z, x \le z).$$

Let $\mathcal{O} = x \prec y \prec z$. We obtain $\mathrm{CAD}_{\varphi(x,y,z)}(\mathcal{O}) = \{0 < x \leq 10 \wedge x \leq y \leq 10 \wedge 0 \leq z \leq x+y,$ $0 < x \leq 10 \wedge 0 \leq y < x \wedge x \leq z \leq 20,\ 10 < x \leq 20 \wedge 0 \leq y \leq 10 \wedge x \leq z \leq 20\}$ and $F_3 = [\![0 < x \leq 10 \wedge x \leq y \leq 10 \wedge 0 \leq z \leq x+y]\!] \cdot 1 + [\![0 < x \leq 10 \wedge 0 \leq y < x \wedge x \leq z \leq 20]\!] \cdot 1$ $+ [\![10 < x \leq 20 \wedge 0 \leq y \leq 10 \wedge x \leq z \leq 20]\!] \cdot 1$.

4.2 Elimination of Randomized Quantifiers

For a randomized quantifier $\mathfrak{U}_{\pi_k} x_k$, the elimination process leverages the fact that integration does not require comparing the functional values of f_i across different cells. Since each c_i is a cylindrical cell with respect to $\mathcal{O}$, the cylindricality property of Definition 2 ensures that the cell itself provides the precise integration bounds $(\ell(x_1, \ldots, x_{k-1}), u(x_1, \ldots, x_{k-1}))$ for the variable x_k.

Specifically, for a randomized quantifier $\mathfrak{U}_{\pi_k} x_k$, each branch (c_i, f_i) defines a cylindrical region over the variables $(x_1, \ldots, x_k)$, where f_i has a fixed semi-algebraic form. For each such cell, the cylindricality property of the CAD with respect to $\mathcal{O}$ ensures that the projection onto the x_k-dimension yields a single interval

$$x_k \in (\ell(x_1, \ldots, x_{k-1}), u(x_1, \ldots, x_{k-1})),$$

where ℓ and u are semi-algebraic functions determined by the CAD. The contribution of the cell is obtained by integrating f_i with the density function:

$$\int_{\ell(x_1,\ldots,x_{k-1})}^{u(x_1,\ldots,x_{k-1})} f_i(x_1, \ldots, x_k)\, \pi_k(x_k)\, dx_k. \tag{12}$$

In this work, we focus on instances for which the integrals are well-defined and admit symbolic representations compatible with our semi-algebraic framework. Summing the contributions of all cells produces a new probability expression in terms of variables $(x_1, \ldots, x_{k-1})$.

▶ **Example 9.** Continuing Example 8, we eliminate the quantifiers iteratively. Since

$$\int_0^{x+y} \frac{1}{20}\, dz = \frac{x+y}{20}, \qquad \int_x^{20} \frac{1}{20}\, dz = 1 - \frac{x}{20},$$

it follows that $F_2 = \mathfrak{U}_\pi z.\, F_3 = [\![0 < x \leq 10 \wedge x \leq y \leq 10]\!] \cdot \frac{x+y}{20} + [\![0 < x \leq 10 \wedge 0 \leq y < x]\!] \cdot (1 - \frac{x}{20}) + [\![10 < x \leq 20 \wedge 0 \leq y \leq 10]\!] \cdot (1 - \frac{x}{20})$.

4.3 Elimination of Strategic Quantifiers

In contrast to randomized quantifiers, eliminating existential $(\exists x_k)$ and universal $(\forall x_k)$ quantifiers requires computing the supremum or infimum of f_i within the cylindrical cell c_i. Since the value of the function f_i may vary over the cell, we must track the range of f_i. To achieve this, we introduce an auxiliary variable θ to represent the value of f_i, and construct an extended CAD for each branch (c_i, f_i).

Since f_i is semi-algebraic, the relation $\theta = f_i(x_1, \ldots, x_k)$ is representable by a quantifier-free NRA formula. Hence, the extended CAD is constructed from the polynomials defining c_i together with those appearing in this representation, using an extended variable ordering $\mathcal{O}_\theta = x_1 \prec \cdots \prec x_{k-1} \prec \theta \prec x_k$. We write $\theta = f_i$ as shorthand for this NRA representation. This specific ordering ensures that the range of possible values for f_i is captured as an interval for θ, allowing us to identify the supremum or infimum relative to the remaining variables.

Specifically, for existential $(\exists x_k)$ and universal $(\forall x_k)$ quantifiers, a CAD is constructed over the variables $(x_1, \ldots, x_{k-1}, \theta, x_k)$ using the polynomials defining c_i and those in $\theta = f_i$. The specific variable ordering $\mathcal{O}_\theta$ is crucial to the elimination process. By placing θ before

x_k, the CAD algorithm treats x_k as the variable to be projected away, while θ remains in the resulting lower-dimensional space. The cylindricality property then ensures that the projection of the constraint $f_i(x_1, \ldots, x_k) - \theta = 0$ onto the $(x_1, \ldots, x_{k-1}, \theta)$-space yields a set of cells in which the bounds of θ are expressed by semi-algebraic functions of the preceding variables $(x_1, \ldots, x_{k-1})$. This structure guarantees that the range of f_i is captured as a semi-algebraically described interval for θ, allowing the direct extraction of the supremum or infimum. Each resulting CAD cell contained within c_i determines a feasible range for the auxiliary variable θ of the form

$$\theta \in (\ell(x_1, \ldots, x_{k-1}), u(x_1, \ldots, x_{k-1})),$$

where ℓ and u are semi-algebraic functions derived from the CAD.

To eliminate θ, we select the upper bound $u(x_1, \ldots, x_{k-1})$ in the case of $\exists x_k$, corresponding to the supremum of f_i and the lower bound $\ell(x_1, \ldots, x_{k-1})$ in the case of $\forall x_k$, corresponding to the infimum of f_i. The selected bound determines the value of the probability expression on the projection of the cell onto $(x_1, \ldots, x_{k-1})$. Summing the contributions of all relevant cells yields a probability expression for F_{k-1} as a finite case distinction over $(x_1, \ldots, x_{k-1})$.

A probability expression represents only the branches retained during its construction. Points satisfying none of the branch guards are therefore represented implicitly with value 0. Although such zero-valued regions can be ignored under integration and supremum, they cannot be ignored when computing an infimum.

For the elimination of a universal quantifier $\forall x_k$, Algorithm 1 first checks, for each fixed assignment to $x_1, \ldots, x_{k-1}$, whether the branch guards cover the entire domain of x_k, that is, whether

$$\forall x_k \in \mathrm{dom}(x_k).\ \bigvee_i \phi_{c_i}(x_1, \ldots, x_k)$$

holds. If this condition does not hold, the formula-induced CAD for

$$D_k(x_1, \ldots, x_k) \wedge \bigwedge_i \neg\phi_{c_i}(x_1, \ldots, x_k)$$

is used to identify its projection onto $x_1, \ldots, x_{k-1}$. The universally quantified expression has value 0 on this projection. Elsewhere, the lower-bound construction described above applies. In Algorithm 1, ProjectUncoveredRegion(F_k, x_k) returns Z, the projection of the portion of $\mathrm{dom}(x_k)$ not covered by any branch guard. Z equals $\emptyset$ if the guards cover the entire domain. During merging, the resulting expression is assigned value 0 on the projection region Z.

Section 4.3 can be concluded by the following lemma.

▶ **Lemma 10.** *Let*

$$F(x_1, \cdots, x_{k-1}) = Q x_k\ [\![\phi]\!] \cdot f, \quad Q \in \{\exists, \forall\}, \quad \mathcal{O}_\theta = x_1 \prec \cdots \prec x_{k-1} \prec \theta \prec x_k.$$

Suppose a cell in $\mathrm{CAD}_{\phi \wedge \theta = f}(\mathcal{O}_\theta)$ *is characterized by*

$$\phi'(x_1, \ldots, x_{k-1}) \ \wedge\ \ell(x_1, \ldots, x_{k-1}) \le \theta \le u(x_1, \ldots, x_{k-1}) \ \wedge\ \psi'(x_1, \ldots, x_{k-1}, \theta, x_k).$$

For the universal case, let $\phi_Z(x_1, \ldots, x_{k-1})$ *characterize the projection of* $D_k(x_1, \ldots, x_k) \wedge \neg\phi(x_1, \ldots, x_k)$ *onto* $x_1, \ldots, x_{k-1}$*. Then the contribution induced by this cell is*

$$\begin{cases} [\![\phi']\!] \cdot u(x_1, \ldots, x_{k-1}) & \text{if } Q = \exists, \\ [\![\phi' \wedge \neg\phi_Z]\!] \cdot \ell(x_1, \ldots, x_{k-1}) & \text{if } Q = \forall. \end{cases}$$

4.4 Merging of Equivalent Regions

In rebuilding the probability expression, distinct cells in $\mathbb{R}^k$ may have the same projection onto $\mathbb{R}^{k-1}$. In such cases, multiple branches of the probability expression will share the same guard, but may be associated with different probability functions.

To preserve a canonical finite case distinction, we merge such branches according to the semantics of the eliminated quantifier as follows.

- For an existential quantifier ($\exists x_k$), we take the maximum of the associated values, corresponding to the supremum operation.
- For a universal quantifier ($\forall x_k$), we take the minimum of the associated values, corresponding to the infimum operation.
- For a randomized quantifier ($\mathfrak{U}_{\pi_k} x_k$), we sum the associated contributions, in accordance with the linearity of integration.

This merging step ensures that the resulting probability expression remains a well-defined finite case distinction with mutually disjoint guards. Although the minimum and maximum of semi-algebraic functions are themselves semi-algebraic, these operators can be eliminated to maintain a standard piecewise representation. To do so, one can use CAD to partition the domain according to the relative ordering of the functions involved. By constructing a CAD sign-invariant for the differences $f_i - f_j$, the space is divided into cells where the inequality $f_i \geq f_j$ (or $f_j \geq f_i$) holds constantly, allowing the minimum or maximum to be replaced by the appropriate function on each cell.

▶ **Example 11.** Continuing Example 9, to eliminate the universal quantifier, we compute extended CADs for the branches of F_2. For the first branch (c_1, f_1), the decomposition yields

$$\mathrm{CAD}_{\phi_{c_1} \wedge (f_1 = \theta)}(\mathcal{O}_\theta) = \{c_1'\}$$

for $\phi_{c_1'} = (0 < x \leq 10 \wedge \frac{x}{10} \leq \theta \leq \frac{x+10}{20} \wedge y = 20\,\theta - x)$. Applying the same construction to branches (c_2, f_2) and (c_3, f_3), we obtain new cells c_2' and c_3' with $\phi_{c_2'} = (0 < x \leq 10 \wedge \theta = 1 - \frac{x}{20} \wedge 0 \leq y < x)$ and $\phi_{c_3'} = (10 < x \leq 20 \wedge \theta = 1 - \frac{x}{20} \wedge 0 \leq y \leq 10)$. By the extended ordering $\mathcal{O}_\theta$, we identify the lower bounds of θ as functions of x, yielding the intermediate probability expression

$$[\![0 < x \leq 10]\!] \cdot \frac{x}{10} + [\![0 < x \leq 10]\!] \cdot (1 - \frac{x}{20}) + [\![10 < x \leq 20]\!] \cdot (1 - \frac{x}{20}).$$

The first two branches have the same projected guard $0 < x \leq 10$, so they are merged by taking the minimum of their associated values:

$$\forall y.\, F_2 = [\![0 < x \leq 10]\!] \cdot \min(\frac{x}{10}, 1 - \frac{x}{20}) + [\![10 < x \leq 20]\!] \cdot (1 - \frac{x}{20})$$

If it is necessary to eliminate min operators, CAD may also be employed to systematically remove such operators. For $\phi = 0 < x \leq 10$, $f_1 = \frac{x}{10}$, and $f_2 = 1 - \frac{x}{20}$, we have

$$\mathrm{CAD}_{\phi \wedge (f_1 \leq f_2)}(x) = \{0 < x \leq \frac{20}{3}\} \text{ and } \mathrm{CAD}_{\phi \wedge (f_1 > f_2)}(x) = \{\frac{20}{3} < x \leq 10\},$$

yielding

$$F_1 = \forall y.\, F_2 = \left[\!\!\left[0 < x \leq \frac{20}{3}\right]\!\!\right] \cdot \frac{x}{10} + \left[\!\!\left[\frac{20}{3} < x \leq 20\right]\!\!\right] \cdot (1 - \frac{x}{20}).$$

Finally, to eliminate the outermost existential quantifier $\exists x$, we construct extended CADs for the branches (c_1, f_1) and (c_2, f_2) of F_1 with

$$\mathrm{CAD}_{\phi_{c_1} \wedge (f_1 = \theta)}\,(\theta \prec x) = \{0 < \theta \leq \frac{2}{3} \wedge x = 10\,\theta\} \text{ and}$$
$$\mathrm{CAD}_{\phi_{c_2} \wedge (f_2 = \theta)}\,(\theta \prec x) = \{0 \leq \theta < \frac{2}{3} \wedge x = 20 - 20\,\theta\}.$$

To finalize the elimination, we select the upper bounds of θ and merge the branches using the max operator, yielding

$$\exists x.\, F_1 = \max(\frac{2}{3}, \frac{2}{3}) = \frac{2}{3}.$$

4.5 Strategy Extraction

Beyond identifying the bounds for θ, the cylindrical structure of the extended CAD facilitates the explicit synthesis of winning strategies for the existential and universal players. Since x_k is the innermost variable in the ordering $\mathcal{O}_\theta$, each cell $c' \subseteq c_i$ in the $(x_1, \ldots, x_{k-1}, \theta, x_k)$-space provides a semi-algebraic description of x_k as a function of the preceding variables and the auxiliary variable θ. Consequently, the strategy σ_{x_k} is obtained by extracting the x_k-component of the satisfying cells, which may correspond either to a unique algebraic value or to an interval of valid assignments. Although the CAD initially represents x_k as a function of both the preceding variables and the auxiliary variable θ, the final strategy σ_{x_k} is derived by substituting θ with its optimal bound: u for $\exists x_k$ and ℓ for $\forall x_k$. This yields a direct mapping from the assigned variables to the player's move, thereby eliminating the dependence on the auxiliary variable θ. As a result, the solver can output not only the final satisfaction probability but also the corresponding sequence of strategies that achieves it.

▶ **Example 12.** Continuing Example 11, we can extract the strategy for y as

$$\sigma_y(x) = \begin{cases} 20 \cdot \frac{x}{10} - x = x & 0 < x \leq \frac{20}{3} \\ [0, x) & \frac{20}{3} < x \leq 10 \\ [0, 10] & 10 < x \leq 20 \end{cases}$$

and the strategy for x as

$$\sigma_x = 10 \cdot \frac{2}{3} = \frac{20}{3}.$$

4.6 Algorithm Correctness

The correctness of the proposed elimination method is asserted by the following theorem.

▶ **Theorem 13** (Correctness of Quantifier Elimination). *Let $\Phi = Q_1 x_1 \ldots Q_n x_n : \varphi$ be a closed SSMT formula, and F_k be the intermediate probability expression produced after eliminating $Q_n, \ldots, Q_{k+1}$. Assume that all CAD operations are exact and that every intermediate integral remains symbolically representable within the semi-algebraic framework. The recursive elimination of quantifiers in the order of Q_n, …, Q_1 through the construction of extended CADs correctly implements the infimum, supremum, and integration semantics of Definition 6, and the final computed value F_0 equals the satisfaction probability* $\Pr(\Phi)$.

Proof. We prove the claim by backward induction on k, from $k = n$ to $k = 0$. For $k = n$, F_n is constructed from the formula-induced CAD of the quantifier-free matrix φ. It assigns value 1 to the cells satisfying φ and value 0 elsewhere. Hence, for every assignment $\mathbf{a} = (a_1, \ldots, a_n)$, $F_n(\mathbf{a}) = 1$ if $\mathbf{a} \models \varphi$, and $F_n(\mathbf{a}) = 0$ otherwise, which establishes the base case. For the inductive step, assume that F_k correctly represents the semantics of the suffix $Q_{k+1} x_{k+1} \cdots Q_n x_n : \varphi$. We show that eliminating $Q_k x_k$ produces the correct expression F_{k-1}. Let $F_k(x_1, \ldots, x_k) = \sum_i [\![\phi_{c_i}]\!] \cdot f_i$ be a probability expression. By the inductive hypothesis, F_k correctly represents the probability of satisfaction for the remaining suffix. We show the correctness for each quantifier Q_k:

1. Randomized Quantifier ($\mathfrak{U}_{\pi_k} x_k$): According to the semantics, F_{k-1} is the integral of F_k over $\mathrm{dom}(x_k)$ weighted by the density π_k. For each branch (c_i, f_i), the algorithm utilizes the fact that c_i is a cylindrical cell with respect to $\mathcal{O}$. Due to the cylindricality property with respect to $\mathcal{O}$, for any fixed values of $x_1, \ldots, x_{k-1}$, the set of values x_k within the cell forms a single interval $(\ell(x_1, \ldots, x_{k-1}), u(x_1, \ldots, x_{k-1}))$. For any x_k outside this interval, the Iverson bracket $[\![\phi_{c_i}]\!]$ evaluates to zero. Therefore:

$$\int_{\mathrm{dom}(x_k)} [\![\phi_{c_i}]\!] \cdot f_i \cdot \pi_k(v)\, dv = \int_{\ell(x_1,\ldots,x_{k-1})}^{u(x_1,\ldots,x_{k-1})} f_i(x_1, \ldots, x_{k-1}, v) \cdot \pi_k(v)\, dv$$

   The linearity of integration ensures that summing these contributions across the cells resulting from the projection of c_i yields the correct value for F_{k-1}.
2. Existential and Universal Quantifiers ($\exists x_k, \forall x_k$): For each branch (c_i, f_i), the extended CAD is constructed using the polynomials defining the cell c_i and those in $\theta = f_i$. Due to the property of this decomposition with respect to $\mathcal{O}_\theta = x_1 \prec \cdots \prec x_{k-1} \prec \theta \prec x_k$, every resulting cell c' that projects into c_i provides a range for θ. For any fixed values of $x_1, \ldots, x_{k-1}$, this range is an interval (ℓ, u) representing the values attained by f_i within that portion of the cell.
   - If $Q_k = \exists x_k$, the semantics require the supremum of F_k. The upper bound $u(x_1, \ldots, x_{k-1})$ identifies the supremum of the values that f_i can take within the cell.
   - If $Q_k = \forall x_k$, first suppose that the branch guards cover the complete domain of x_k. In this case, the lower bounds obtained from the extended CADs give the infima over the represented cells, and taking their minimum gives the infimum over the complete domain. If the guards do not cover the complete domain, the formula-induced CAD of the uncovered region described in Section 4.3 identifies its projection onto $x_1, \ldots, x_{k-1}$. On this projection, some admissible value of x_k satisfies no branch guard, so $F_k = 0$ there. Since probability values are nonnegative, the infimum is 0; outside this projection, the lower-bound construction applies.

Finally, the merging step combines the values associated with branches whose guards correspond to the same region in $\mathbb{R}^{k-1}$. For universal quantification, the additional zero-valued projection identified in Section 4.3 accounts for portions of the domain not covered by any branch guard. Thus, F_{k-1} agrees with the semantics of $Q_k x_k$ in every case. By backward induction, $F_0 = \Pr(\Phi)$. ◀

5 Extensions

5.1 Encoding of Boolean Variables

Although our problem formulation assumes that all variables range over the real numbers, Boolean variables can be accommodated through a real-valued threshold encoding. Let $\Phi = Q : \varphi$ be an SSMT formula containing a set of variables that are intended to be Boolean. For each such Boolean variable b, we introduce a new real variable x_b with the interval $[0, 1]$ as its domain.

We distinguish the treatment of Boolean variables according to the type of quantifier:

- **Strategic Quantifiers.** For a Boolean variable b quantified by $\forall$ or $\exists$, its Boolean literals are then translated into arithmetic constraints by interpreting literal b as $(x_b \leq \frac{1}{2})$ and literal $\neg b$ as $(x_b > \frac{1}{2})$.
- **Random Quantifiers.** For a Boolean variable quantified as a random variable, denoted $\mathfrak{U}^p b$, we first transform it into a real-valued random variable $x_b \sim \mathcal{U}(0, 1)$. Boolean literals are then translated into arithmetic constraints by interpreting literal b as $(x_b \leq p)$ and literal $\neg b$ as $(x_b > p)$.

Under this encoding, Boolean reasoning is reduced to reasoning over real-valued constraints, allowing the original algorithm and solver to be applied without modification. A proof of the correctness of the encoding is provided in Appendix A.

▶ **Example 14.** The SSMT formula

$$\exists b \in \{\top, \bot\},\ \mathfrak{U} y \sim \mathcal{U}(0,1).\ (b \wedge y > \frac{4}{5}) \vee (\neg b \wedge y < \frac{1}{5}).$$

can be converted to

$$\exists x_b \in [0,1], \mathfrak{U} y \sim \mathcal{U}(0,1).\ (x_b \leq \frac{1}{2} \wedge y > \frac{4}{5}) \vee (x_b > \frac{1}{2} \wedge y < \frac{1}{5}).$$

5.2 Variable Reparameterization

Since our algorithm relies on CAD, we require constraints in SSMT formulas to be semi-algebraic. Nevertheless, under certain conditions, formulas involving non-semi-algebraic expressions can still be handled by reparameterizing quantified variables.

We allow *reparameterization of quantified variables* in SSMT formulas. Specifically, if a quantified variable appears in the matrix only through a derived term, then the quantifier may be reparameterized to range directly over the value of that term. This allows non-semi-algebraic expressions to be removed from the matrix while preserving the semantics of existential and universal quantifiers and remaining within the theory of real closed fields. For randomized quantifiers, however, such a reparameterization is not sound in general, since it changes the induced probability distribution and may require introducing a transformed density function that is not semi-algebraic.

For $\Phi = Q_1 x_1 \in \mathrm{dom}(x_1),\ \ldots,\ Q_i x_i \in \mathrm{dom}(x_i),\ \ldots,\ Q_n x_n \in \mathrm{dom}(x_n) :\ \varphi$, assume that $Q_i \in \{\exists, \forall\}$, and that there exists a function $f(x_1, \ldots, x_{i-1}, x_i)$ and a quantifier-free formula φ' such that

$$\varphi \ \equiv\ \varphi'(x_1, \ldots, x_{i-1}, f(x_1, \ldots, x_{i-1}, x_i)),$$

where the variable x_i does not occur free in φ' except through the argument $f(\cdot)$; that is, the matrix depends on x_i only via the derived term $f(x_1, \ldots, x_{i-1}, x_i)$. We introduce a fresh variable y and define the induced domain

$$\mathrm{dom}(y) \coloneqq f\big(\mathrm{dom}(x_i)\big).$$

We then define the reparameterized formula

$$\begin{aligned}\Phi' = {} & Q_1 x_1 \in \mathrm{dom}(x_1),\ \ldots,\ Q_{i-1} x_{i-1} \in \mathrm{dom}(x_{i-1}),\\ & Q_i y \in \mathrm{dom}(y),\ \ldots,\ Q_n x_n \in \mathrm{dom}(x_n) :\ \varphi'(x_1, \ldots, x_{i-1}, y, \ldots, x_n).\end{aligned}$$

This reparameterization is sound for existential and universal quantifiers because the matrix does not distinguish between different values of the quantified variable x_i that yield the same value of the derived term. Consequently, quantifying directly over the range of that term preserves the semantics of the original formula while simplifying the structure of the matrix.

▶ **Example 15.** An SSMT formula

$$\forall x \in \mathbb{R},\ \mathfrak{U} y \sim \mathcal{U}(0,1).\ \sin(x) \geq y$$

can be reformulated as

$$\forall x' \in [-1, 1],\ \mathfrak{U} y \sim \mathcal{U}(0,1).\ x' \geq y,$$

which no longer contains non-semi-algebraic functions.

While randomized quantifiers remain restricted to densities whose integrals admit representations compatible with our semi-algebraic framework, the reparameterization technique above provides a way to handle certain non-semi-algebraic expressions. Non-semi-algebraic distributions, such as the Gaussian distribution, remain a significant challenge due to their transcendental nature. However, our reparameterization technique in Section 5.2 demonstrates how certain non-semi-algebraic components can be mapped into a semi-algebraic representation, providing a potential pathway for future extensions to a broader class of probability distributions.

6 Experimental Evaluation

6.1 Implementation Details

Our SSMT framework was implemented using a combination of specialized symbolic and numerical computing tools. Specifically, we leveraged `Wolfram|Alpha` [15] as the primary engine for Cylindrical Algebraic Decomposition (CAD) and `Z3` [7] as the underlying SMT solver to perform satisfiability checks. Additionally, `SymPy` [16] was employed for high-precision numerical and symbolic calculations throughout the integration process.

A key component of our implementation is to take advantage of the `Cylindrical Decomposition` function in `Wolfram|Alpha` to compute the formula-induced CAD, defined in Definition 3. Specifically, given a Boolean+NRA formula and a predefined variable ordering, the engine identifies the set of cylindrical cells within which the formula consistently evaluates to TRUE, providing the necessary geometric foundation for the subsequent quantifier elimination steps.

To enhance computational efficiency, we incorporated an optimization strategy designed to minimize redundant CAD invocations, which are typically the most expensive component of the process. Before invoking the CAD procedure, our solver heuristically employs `Z3` to check the satisfiability of some Boolean formulas. If the SMT solver determines that a formula is unsatisfiable ($\bot$), the framework skips the corresponding CAD computation for that branch, thereby significantly reducing the overall execution time.

6.2 Benchmark Evaluation

We evaluated our SSMT framework using a curated suite of benchmarks categorized into three distinct groups, designed to test the correctness, expressiveness, and practical applicability of our solver.

6.2.1 Group 1: Baseline Synthetic Formulas

To test the logical and numerical robustness of our framework, we collected a suite of synthetic benchmarks. These instances serve as a baseline regression test to ensure that the CAD-based elimination procedure correctly handles diverse structural configurations, such as various quantifier prefixes and variable-dependent domains, before evaluating high-complexity strategic models.

6.2.2 Group 2: Strategic Economic Models

To assess the framework's ability to model high-level strategic interactions, we utilized benchmarks representing strategic economic competition. These instances model a duopoly price-setting game where two competing agents (Leader and Follower) must choose optimal

strategies under environmental uncertainty. Such models require the solver to handle interdependent existential, universal, and randomized quantifiers within a profit-maximizing context, providing a rigorous test for our solver's support for full quantifier interplay.

6.2.3 Group 3: Probabilistic Program Verification

Finally, to demonstrate the broader utility of our framework in software analysis, we incorporated benchmarks derived from the `ProbInG` suite [17, 21]. We significantly adapted the original probabilistic programs to leverage the expressive power of our SSMT framework in two key aspects:

- Domain Transformation. Since our framework operates over real closed fields to ensure exact algebraic reasoning, we relaxed the discrete variables found in certain `ProbInG` programs into the real domain. This allows for the application of CAD-based symbolic integration to compute exact satisfaction probabilities for non-linear real arithmetic (NRA) properties.
- Introduction of Strategic Nondeterminism. We introduced nondeterministic inputs (modeled via `nondet()`) into the programs to represent strategic environmental factors or controlled system parameters. This transformation enables the use of existential ($\exists$) and universal ($\forall$) quantifiers to model adversarial or cooperative interactions within the stochastic execution.

From these adapted programs, we extracted six representative system characteristics and reformulated them into NRA-based SSMT formulas. Since the original benchmarks contain loops, we performed finite loop unrolling to translate them into quantifier-free matrices. While specialized verification tools such as `Polar` [18] or `Prodigy` [12] can compute exact moments or distributions for certain programs, our framework is uniquely designed to handle the strategic interplay of existential and universal quantifiers within an SSMT setting, showcasing the effectiveness of combining strategic quantification with stochastic reasoning.

▶ **Example 16.** Given the probabilistic program

```
x = 0
a = nondet()
assert 0 <= a < 1
b = nondet()
assert a < b <= 1
while true:
    y = Uniform(a,b)
    x = x + y
end
```

Assume the input a is controlled by $\exists$ with $a \in [0, 1)$, and b is controlled by $\forall$ with $b \in (a, 1]$. Although the original program contains an unbounded loop, our framework can analyze bounded executions via finite loop unrolling. For example, after unrolling three iterations, the probability of $x \geq 2$ can be encoded as the following SSMT formula:

$$\exists a \in [0, 1)\, \forall b \in (a, 1]\, \mathfrak{U}_\pi y_1,\ \mathfrak{U}_\pi y_2,\ \mathfrak{U}_\pi y_3.\ y_1 + y_2 + y_3 \geq 2$$

where $y_i \in [a, b]$ and $\pi = \frac{1}{b-a}$.

Table 1 Formula and Performance Statistics.

		Formula Statistics		Performance Statistics			
Group	**#Inst**	**#Quant**	**#Cell**	**Time (s)**	**CAD %**	**#CAD**	**#Cell ∃∀**
Base	17	2.29	2.53	10.79	98.80%	3.29	1.76
Game	21	3.00	6.81	33.34	95.14%	12.00	4.71
Prob	6	3.67	14.83	31.35	91.71%	6.50	3.83

6.3 Results and Analysis

Our framework successfully solved all 44 benchmarks (Group 1: 17, Group 2: 21, Group 3: 6) within a 100-second timeout limit. The solver derived exact symbolic probability expressions for all cases, achieving symbolic precision that is unattainable by traditional numerical or sampling-based SSMT solvers.

The experimental data, summarized in Table 1, confirms that Cylindrical Algebraic Decomposition (CAD) is the dominant factor in execution time. We report the **CAD %**, defined as the percentage of total execution time consumed by the CAD procedure. In Group 1, this metric averaged 98.80%, with peak values reaching 99.95% (e.g., `gen1_12`). Similarly, Group 2 showed an average CAD percentage of 95.14%. These high percentages provide strong empirical support for our optimization strategy. Since CAD is the most expensive component, our heuristic use of `Z3` to prune unsatisfiable branches and reduce the number of CAD invocations is critical for scalability.

While quantifier depth and the initial number of satisfying cells are intuitive measures of a formula's complexity, our empirical results show they are inconsistent predictors of total execution time. Quantifier depth alone does not account for the varying computational costs of different quantifier types, as the symbolic integration used for randomized quantifiers ($\mathfrak{R}$) is significantly more efficient than the extended CAD required for strategic quantifiers ($\exists, \forall$). Similarly, the initial cell count, the number of branches in F_n, can be misleading because the integration phase often simplifies the probability expression, merging or eliminating cells before the algorithm reaches the strategic bottleneck.

Consequently, we identify the *Strategic Branching Factor* as the primary determinant of computational cost. This metric is defined as the number of active branches in the probability expression F_k at the moment the algorithm encounters the first strategic quantifier during its innermost-to-outermost elimination process. As shown in Figure 1, runtime correlates strongly with this factor because each strategic branch necessitates the construction of an extended CAD with auxiliary variables, which represents the dominant performance bottleneck of the solver. By focusing on branching at this specific transition point, we filter out the initial decompositions and efficient integration steps, providing a precise measure of the workload passed to the most expensive symbolic procedures.

We additionally conducted limited comparisons with related tools on restricted classes of problems. For pure quantifier elimination instances without randomized quantifiers, `Z3` [7] generally achieved lower runtimes, as expected, since such solvers are specifically optimized for classical quantifier elimination tasks and do not construct the symbolic probability expressions required in our setting.

We also evaluated `Polar` [18] on related probabilistic-program benchmarks as an additional point of comparison. While `Polar` is effective for moment computation and probabilistic inference, it does not support the strategic interaction between existential and universal quantifiers considered in SSMT. These comparisons highlight that the considered approaches are designed for different classes of probabilistic reasoning problems.

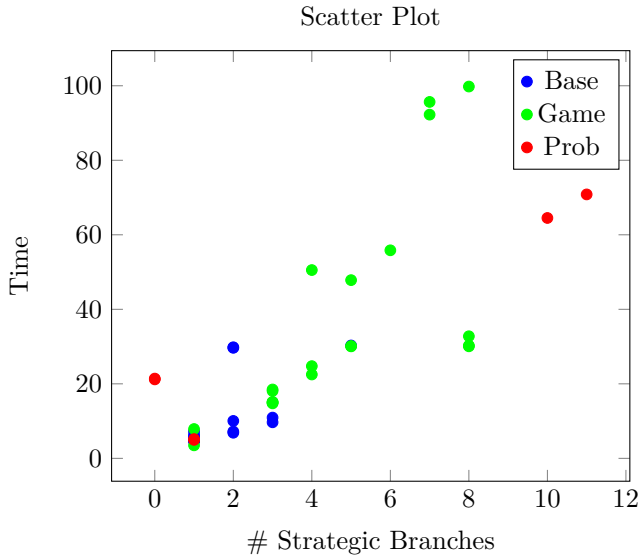

Figure 1 Scatter plot showing the relationship between the number of strategic branches and execution time for the baseline synthetic formulas (Base), strategic economic models (Game), and probabilistic program verification (Prob) benchmarks.

7 Conclusion

We have presented a novel SSMT framework that achieves full quantifier support. By incorporating universal quantification, we have transformed SSMT from a simple reachability tool into a stochastic game solver capable of capturing system performance in adversarial environments or under worst-case perturbations. Based on CAD for symbolic quantifier elimination, the satisfaction probabilities can be computed exactly, in contrast to the inaccuracies inherent in sampling-based approaches. Furthermore, our implementation demonstrates that combining specialized symbolic tools, such as `Wolfram|Alpha` for CAD and `SymPy` for integration, with `Z3`-based pruning, leads to a robust and efficient solver.

For future work, while our current framework is tailored for algebraic density functions and real-valued variables, the variable reparameterization method provides an opportunity to extend to transcendental distributions and non-semi-algebraic constraints.

References

1 S. Akshay, Supratik Chakraborty, Amir Kafshdar Goharshady, R. Govind, Harshit Jitendra Motwani, and Sai Teja Varanasi. Practical approximate quantifier elimination for non-linear real arithmetic. In André Platzer, Kristin Yvonne Rozier, Matteo Pradella, and Matteo Rossi, editors, *Formal Methods*, pages 111–130, Cham, 2025. Springer Nature Switzerland.

2 Saugata Basu. Algorithms in real algebraic geometry: A survey. In Karim Bekka, Goulwen Fichou, Jean-Philippe Monnier, and Ronan Quarez, editors, *Real Algebraic Geometry*, volume 51 of *Panoramas et Synthèses*, pages 107–153. Société Mathématique de France, 2017. URL: `https://smf.emath.fr/node/44883`.

3 Nikolaj S Bjørner and Mikolás Janota. Playing with quantified satisfaction. *LPAR (short papers)*, 35:15–27, 2015. `doi:10.29007/VV21`.

4 Maria Paola Bonacina, Stéphane Graham-Lengrand, and Christophe Vauthier. QSMA: a new algorithm for quantified satisfiability modulo theory and assignment. In *International Conference on Automated Deduction*, pages 78–95. Springer, 2023. `doi:10.1007/978-3-031-38499-8_5`.

5 Bob F. Caviness and Jeremy R. Johnson, editors. *Quantifier Elimination and Cylindrical Algebraic Decomposition*. Texts & Monographs in Symbolic Computation. Springer Vienna, 1 edition, 1998. doi:10.1007/978-3-7091-9459-1.

6 George E. Collins. Quantifier elimination for real closed fields by cylindrical algebraic decompostion. In H. Brakhage, editor, *Automata Theory and Formal Languages*, pages 134–183, Berlin, Heidelberg, 1975. Springer Berlin Heidelberg.

7 Leonardo de Moura and Nikolaj Bjørner. Z3: An efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

8 Martin Fränzle, Holger Hermanns, and Tino Teige. Stochastic satisfiability modulo theory: A novel technique for the analysis of probabilistic hybrid systems. In Magnus Egerstedt and Bud Mishra, editors, *Hybrid Systems: Computation and Control*, pages 172–186, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-78929-1_13.

9 Yang Gao and Martin Fränzle. A solving procedure for stochastic satisfiability modulo theories with continuous domain. In Javier Campos and Boudewijn R. Haverkort, editors, *Quantitative Evaluation of Systems*, pages 295–311, Cham, 2015. Springer International Publishing. doi:10.1007/978-3-319-22264-6_19.

10 Yang Gao and Martin Franzle. CSiSAT: A satisfiability solver for SMT formulas with continuous probability distributions. In *2016 International Workshop on Symbolic and Numerical Methods for Reachability Analysis (SNR)*, pages 1–6, 2016. doi:10.1109/SNR.2016.7479382.

11 Jaakko Hintikka. Game-theoretical semantics: Insights and prospects. *Notre Dame Journal of Formal Logic*, 23(2):219–241, 1982. doi:10.1305/ndjfl/1093883627.

12 Lutz Klinkenberg, Christian Blumenthal, Mingshuai Chen, Darion Haase, and Joost-Pieter Katoen. Exact bayesian inference for loopy probabilistic programs using generating functions. *Proc. ACM Program. Lang.*, 8(OOPSLA1), 2024. doi:10.1145/3649844.

13 Donald E. Knuth. Two notes on notation. *The American Mathematical Monthly*, 99(5):403–422, 1992. URL: http://www.jstor.org/stable/2325085.

14 Michael L. Littman, Stephen M. Majercik, and Toniann Pitassi. Stochastic Boolean satisfiability. *Journal of Automated Reasoning*, 27(3):251–296, 2001. doi:10.1023/A:1017584715408.

15 Wolfram Alpha LLC. Wolfram|Alpha. https://www.wolframalpha.com/, 2026. Accessed: 2026-02-04.

16 Aaron Meurer, Christopher Smith, Mateusz Paprocki, Ondřej Čertík, Sergey Kirpichev, Matthew Rocklin, AMiT Kumar, Sergiu Ivanov, Jason Moore, Sartaj Singh, Thilina Rathnayake, Sean Vig, Brian Granger, Richard Muller, Francesco Bonazzi, Harsh Gupta, Shivam Vats, Fredrik Johansson, Fabian Pedregosa, and Anthony Scopatz. SymPy: Symbolic computing in python, June 2016. doi:10.7287/peerj.preprints.2083.

17 Marcel Moosbrugger. *Automated analysis of probabilistic loops*. PhD thesis, Technische Universität Wien, 2024.

18 Marcel Moosbrugger, Julian Müllner, Ezio Bartocci, and Laura Kovács. *Polar: An Algebraic Analyzer for (Probabilistic) Loops*, pages 179–200. Springer Nature Switzerland, November 2024. doi:10.1007/978-3-031-75783-9_8.

19 Charlie Murphy and Zachary Kincaid. Quantified linear arithmetic satisfiability via fine-grained strategy improvement. In *International Conference on Computer Aided Verification*, pages 89–109. Springer, 2024. doi:10.1007/978-3-031-65627-9_5.

20 Christos H. Papadimitriou. Games against nature. *Journal of Computer and System Sciences*, 31(2):288–301, 1985. doi:10.1016/0022-0000(85)90045-5.

21 ProbInG Lab, TU Wien. Probing benchmark suite for probabilistic program analysis, 2024. Online; accessed 9 March 2026. URL: https://probing-lab.github.io/benchmarks/.

22 Markus Schweighofer. Real algebraic geometry, positivity and convexity. arXiv preprint arXiv:2205.04211, 2022. doi:10.48550/arXiv.2205.04211.

23 Tino Teige and Martin Fränzle. Stochastic satisfiability modulo theories for non-linear arithmetic. In Laurent Perron and Michael A. Trick, editors, *Integration of AI and OR*

Techniques in Constraint Programming for Combinatorial Optimization Problems, pages 248–262, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. doi:10.1007/978-3-540-68155-7_20.

A Correctness Proof of the Boolean Encoding

Recall that we replace a Boolean variable b in an SSMT formula Φ with a real variable $x_b \in [0,1]$ such that every appearance of literal b (respectively, $\neg b$) in the matrix of Φ is replaced with $x_b \leq \tau_b$ (respectively, $x_b > \tau_b$), where

$$\tau_b = \begin{cases} 1/2, & \text{if } b \text{ is quantified by } \exists \text{ or } \forall, \\ p, & \text{if } b \text{ is quantified by } \mathfrak{U}^p. \end{cases}$$

▶ **Theorem 17** (Semantic Preservation of Boolean Encoding). *Let Φ be an SSMT formula whose variables are partitioned into real-valued variables and Boolean variables. Assume that Boolean variables appear as literals, b or $\neg b$, in the Boolean structure of the quantifier-free matrix, and do not appear elsewhere (e.g., in arithmetic terms, polynomial constraints, probability density functions, or quantifier-domain bounds). Let* $\mathrm{enc}(\Phi)$ *denote the real-valued SSMT formula obtained via the threshold encoding described above. Then,*

$$\Pr_{\mathsf{Bool}}(\Phi) = \Pr_{\mathsf{Real}}(\mathrm{enc}(\Phi)).$$

Proof. We prove the claim by induction on the length of the quantifier prefix. For the base case (quantifier-free formulas), fix an assignment to all real-valued variables. The encoding does not modify any arithmetic atoms involving only real-valued variables. For every Boolean variable b, the encoding partitions $[0,1]$ into two regions $T_b = [0, \tau_b]$ and $F_b = (\tau_b, 1]$. By construction, b evaluates to TRUE exactly on T_b, and $\neg b$ evaluates to TRUE exactly on F_b. Hence, under any fixed assignment to the real-valued variables, the encoded quantifier-free matrix has the same truth value as the original Boolean matrix under the corresponding Boolean assignment induced by the partition.

For the inductive step, assume the claim holds for all suffix formulas with strictly fewer quantifiers, and consider a formula with leading Boolean quantifier $Qb.\Psi$. For the case of $Q = \exists$, in the encoding, the Boolean variable b corresponds to a partition of $[0,1]$ into T_b and F_b, and the encoded formula is constant on each region. Therefore,

$$\sup_{x_b \in [0,1]} \Pr_{\mathsf{Real}}(\mathrm{enc}(\Psi)) = \max\{\Pr_{\mathsf{Bool}}(\Psi[\top/b]), \Pr_{\mathsf{Bool}}(\Psi[\bot/b])\},$$

which coincides with the Boolean semantics of $\exists b.\Psi$.

For the case of $Q = \forall$, the argument is analogous, replacing $\sup$ / $\max$ with $\inf$ / $\min$, which agrees with the Boolean semantics of $\forall b.\Psi$.

For the case of $Q = \mathfrak{U}^p$, the encoding interprets b as a random variable $x_b \sim \mathcal{U}(0,1)$, with $T_b = [0,p]$ and $F_b = (p,1]$. By the induction hypothesis,

$$\begin{aligned} \int_0^1 \Pr_{\mathsf{Real}}(\mathrm{enc}(\Psi))\, dx_b &= \int_0^p \Pr_{\mathsf{Bool}}(\Psi[\top/b])\, dx_b + \int_p^1 \Pr_{\mathsf{Bool}}(\Psi[\bot/b])\, dx_b \\ &= p \Pr_{\mathsf{Bool}}(\Psi[\top/b]) + (1-p) \Pr_{\mathsf{Bool}}(\Psi[\bot/b]), \end{aligned}$$

which is exactly the Boolean semantics of $\mathfrak{U}^p b.\Psi$.

For the case of real-valued quantifiers, quantifiers over real variables are unaffected by the Boolean encoding, and thus the claim follows directly from the induction hypothesis. Therefore, the encoding preserves the satisfaction probability of the formula. ◀