Automated Safety Verification of Posterior Distributions of Probabilistic Programs

Kazuki Watanabe[1,2] , **Hiroshi Unno**[3]
[1]National Institute of Informatics, Tokyo
[2]The Graduate University for Advanced Studies (SOKENDAI), Hayama
[3]Tohoku University, Sendai
kazukiwatanabe@nii.ac.jp, hiroshi.unno@acm.org

Abstract

Ensuring the safety of probabilistic systems is a central challenge in formal verification. We propose an automated refinement technique for verifying a safety property of posterior distributions in probabilistic programs. Our approach builds on the counterexample-guided abstraction refinement (CEGAR) framework and exploits the duality in convex optimisation and the adequacy of predicate-transformer semantics in probabilistic settings. We implement the technique and evaluate its effectiveness through preliminary experiments.

1 Introduction

Uncertainty is ubiquitous in real-world systems, and probabilistic behaviours are among the most challenging sources of such uncertainty. To model systems or behaviours that involve probability, *probabilistic programs* have proven effective for decades [van de Meent *et al.*, 2018; Barthe *et al.*, 2020; Gordon *et al.*, 2014]. As surveyed in [van de Meent *et al.*, 2018], probabilistic programming has enabled applications such as constrained procedural graphics [Ritchie *et al.*, 2015] and recursive multi-agent reasoning [Goodman and Stuhlmüller, 2014]. A variety of probabilistic programming languages have been developed to address different real-world needs, including Church [Goodman *et al.*, 2008], Anglican [Wood *et al.*, 2014; Tolpin *et al.*, 2016], and Pyro [Bingham *et al.*, 2019].

Automated verification of probabilistic programs has become a central topic in the formal verification of probabilistic systems. In particular, numerous techniques have been proposed for verifying almost-sure termination and quantitative bounds on expected run-time or outcomes (e.g. [Batz *et al.*, 2023; Ngo *et al.*, 2018; Bao *et al.*, 2025; Chakarov and Sankaranarayanan, 2013; Chatterjee *et al.*, 2020; Li *et al.*, 2025]). It is worth noting that verifying almost-sure termination is, in general, undecidable [Kaminski and Katoen, 2015], highlighting the inherent difficulties and challenges in the verification of probabilistic programs.

In this paper, we address a verification problem that has received comparatively little attention in the literature: establishing a safety property over posterior distributions of probabilistic programs. We illustrate a motivating example below.

Motivating Example. We consider two robots with identical architectures that independently perform autonomous exploration. Their behaviours are modelled by the following probabilistic program. Let C denote the following program.

```
b := true, t := 0
while (b) {
   { D; b := false } [1/100] {t += 1}
}
```

where the program D is given by

```
{ {teamA := true} [1/10] {skip} } □ {
    {teamB := true} [1/10] {skip} }
```

Each robot continues exploring until it encounters an issue; for simplicity, we assume that such an issue occurs with probability $1/100$ in each iteration. Once a robot detects an issue and terminates its exploration, it may require manual support from either Team A or Team B, with probability $1/10$, depending on its assigned task. This uncertainty is modelled via nondeterminism, resolved in an adversarial (demonic) manner. Crucially, the two robots share the task but operate independently, regardless of what the task is.

Now suppose that the two robots indeed share the same task and operate independently. Although both robots may request support from the same team, the team cannot serve both simultaneously due to the cost of assistance. Therefore, we would like to ensure that the probability that both robots request help from the same team remains reasonably small. This situation can be captured by the terminating probabilities $P^{\sigma}_{\mathrm{teamA}}$ and $P^{\sigma}_{\mathrm{teamB}}$, where σ denotes a *scheduler* that decides the task assignment by resolving the nondeterminism, and $P^{\sigma}_{\mathrm{teamA}}$ (resp. $P^{\sigma}_{\mathrm{teamB}}$) is the probability that a robot terminates by requesting support from Team A (resp. Team B) under the scheduler σ. Given a threshold $\theta > 0$, the verification task asks whether

$$\sup_{\sigma} (P^{\sigma}_{\mathrm{teamA}})^2 + (P^{\sigma}_{\mathrm{teamB}})^2 \leq \theta.$$

Since the program models the behaviour of a single robot, we take the square of each probability to account for two independent robots.

In this example, by a simple calculation we obtain

$$\sup_{\sigma} (P^{\sigma}_{\mathrm{teamA}})^2 + (P^{\sigma}_{\mathrm{teamB}})^2 = 1/100$$

from the initial state in which both teamA and teamB are assigned false.

Challenge. While the above example can be verified by hand due to its simplicity, it raises several questions when we generalize the underlying safety problem. In particular, given a program and a safe region $R \subseteq [0,1]^n$, we ask whether, regardless of how nondeterminism is resolved by a scheduler σ, the resulting terminating probabilities $P_1^\sigma, \ldots, P_n^\sigma$ (from an initial state) to designated terminal states, which we collectively call the *posterior distribution*, always lie within R, that is,

$$(P_1^\sigma, \ldots, P_n^\sigma) \in R, \text{ for any scheduler } \sigma.$$

First, formulating this problem in a suitable way while keeping it tractable is non-trivial. Allowing an arbitrary R would render the problem unsolvable in practice, so we must identify a reasonable class for R to preserve tractability. Moreover, a naïve approach would require enumerating all posterior distributions induced by schedulers—an infeasible strategy, as the scheduler space is typically infinite. To the best of our knowledge, no existing work provides an automated verification method for this problem in probabilistic programs, including the motivating example above.

Our Approach. In this paper, we first formulate the verification problem under certain assumptions on the safe region R. Specifically, we assume that R is a downward-closed convex set of the form

$$R \coloneqq \{\mathbf{p} \in [0,1]^n \mid f(\mathbf{p}) \leq \theta\},$$

where $\theta \geq 0$ is a threshold and $f\colon [0,1]^n \to \mathbb{R}_{\geq 0}$ is a polynomial with rational coefficients. The motivating example corresponds to the special case $f(p_1, p_2) \coloneqq p_1^2 + p_2^2$.

We then propose a novel automated verification method for this problem. Specifically, we develop a *counterexample-guided abstraction refinement (CEGAR)* framework [Ball and Rajamani, 2002; Clarke *et al.*, 2003] to verify it effectively.

Our approach maintains an abstraction U, represented as a convex polytope obtained as a finite intersection of half-spaces, such that

$$\{(P_1^\sigma, \ldots, P_n^\sigma) \mid \sigma \text{ is a scheduler}\} \subseteq U.$$

To resolve the safety problem, it suffices to show that $U \subseteq R$. We prove that, under our choice of U, checking $U \subseteq R$ is feasible due to topological properties of U and R.

A CEGAR loop requires refinement via counterexamples; in our setting, refinement is triggered when $U \not\subseteq R$. Since U is a convex polytope, we show that counterexamples can be extracted from its vertices. We then refine U to a strictly smaller polytope $U' \subsetneq U$ by adding a new half-space induced by such a counterexample. This half-space is derived by synthesizing a suitable invariant, which can be obtained symbolically by existing solvers (e.g. [Chatterjee *et al.*, 2025]).

We implement our CEGAR loop using well-maintained existing solvers, including Z3 [de Moura and Bjørner, 2008], PolyQEnt [Chatterjee *et al.*, 2025], and Apron [Jeannet and Miné, 2009]. We evaluate the effectiveness of our approach through preliminary experiments.

Structure. In §2, we present preliminaries on probabilistic programs and formally define our verification problem. In §3, we illustrate the key ideas and observations underlying our approach. There, adequacy [Batz *et al.*, 2024] with respect to predicate-transformer semantics [McIver and Morgan, 2005] and the Fenchel-Moreau duality are recalled as crucial ingredients. In §4, we describe our algorithm in detail and establish desirable properties such as progress and soundness. In §5, we report preliminary experimental results obtained based on our implementation. Related work is discussed in §6, and we conclude in §7.

Notation. With a slight abuse of notation, for vectors $\mathbf{p}, \mathbf{q} \in [0,1]^n$ we write $\mathbf{p} \cdot \mathbf{q}$ for their inner product. We write $[m]$ for the set $\{1, \ldots, m\}$ for a natural number m.

2 Preliminaries and Problem Setting

In this section, we introduce our target language, and formalize the verification problem of interest.

2.1 Target Language

Our target language is the *probabilistic guarded command language (pGCL)* [McIver and Morgan, 2005], a simple imperative language. Formally, we fix a countable set of *variables* $\mathrm{Vars} \coloneqq \{x, y, z, \ldots\}$, and define the countably infinite set of *states* by $\mathrm{St} \coloneqq \{s\colon \mathrm{Vars} \to \mathbb{N} \mid \text{the support of } s \text{ is finite}\}$. Note that restricting variables to take non-negative values is a standard assumption (e.g. [Batz *et al.*, 2023]) and it does not reduce expressiveness [Batz *et al.*, 2021]. A *probabilistic program* C is then inductively given by

$$C ::= \mathrm{skip} \mid x \coloneqq e \mid C; C \mid \{C\}[p]\{C\} \mid \{C\}\square\{C\} \\ \mid \text{if } (\phi)\ \{C\} \text{ else } \{C\} \mid \text{while } (\phi)\ \{C\},$$

where 1) $e\colon \mathrm{St} \to \mathbb{Q}_{\geq 0}$ is a *linear arithmetic expression* (such as x and $x + r$ with $r \in \mathbb{R}_{\geq 0}$); 2) $\phi \subseteq \mathrm{St}$ is a predicate that is defined by inequalities, negation, and conjunction; and 3) p is a probability $p \in \mathbb{Q}_{\geq 0} \cap [0,1]$. Here, $\{C\}[p]\{C\}$ and $\{C\}\square\{C\}$ denote probabilistic and nondeterministic branching, respectively. See, e.g., [Batz *et al.*, 2023; Batz *et al.*, 2024] for the full definition of pGCL. As a syntax sugar, we write $p \cdot C_1 + (1-p) \cdot C_2$ for $\{C_1\}[p]\{C_2\}$ and $\square\,\{C_1\}\,\{C_2\}$ for $\{C_1\}\square\{C_2\}$.

2.2 Operational Semantics

For the semantics of pGCL, we employ the operational semantics based on Markov decision processes (MDPs) [Puterman, 1994], following the formulation of [Gretz *et al.*, 2014] in the style of [Plotkin, 2004]. The key idea is that program computations (or executions) are modeled as probabilistic transitions in an MDP, while nondeterministic choices are represented as nondeterministic choices of actions. Program termination corresponds to reaching designated target states in the MDP. We also refer to [Batz *et al.*, 2024] for a self-contained reference on this operational semantics.

We recall standard preliminaries on MDPs. An MDP M is a tuple (Q, T, A, P) consisting of a countable set of *positions* Q, a subset of *terminal positions* $T \subseteq Q$, a finite set of *actions* A, and a *transition probability function* $P\colon Q \times A \times Q \to$

$[0,1]$ satisfying $\sum_{q' \in Q} P(q,a,q') = 1$ for all $q \in Q$ and $a \in A$. We further assume that terminal positions are sinks, i.e. $P(t,a,t) = 1$ for any $t \in T$ and $a \in A$.

Nondeterminism in pGCL is modeled by action choices, which are resolved by *schedulers*. Formally, a scheduler is a function $\sigma \colon Q^+ \to A$. Given a scheduler σ and a path $q_1 \cdots q_m$ of positions, the *path probability* is defined as

$$\mathbb{P}^\sigma(q_1 \cdots q_m) \coloneqq \prod_{i=1}^{m-1} P(q_i, \sigma(q_1 \cdots q_i), q_{i+1}).$$

Finally, given an initial position $q \in Q$ and a scheduler σ, the *terminating probability* on a target position $t \in T$ is defined by

$$\mathbb{TP}^\sigma_{q,t} \coloneqq \sum_{q \cdots q_m \cdot t} \mathbb{P}^\sigma(q \cdots q_m \cdot t),$$

where the summation ranges over paths $q \cdots q_m \cdot t$ such that none of $q, \ldots, q_m$ are terminal.

Following [Gretz *et al.*, 2014], a program C induces an MDP M^C in which each position corresponds to a configuration (s, C') consisting of a state s and a program C'. Each terminal position of M^C is in bijection with a state s, and we write t_s for the terminal position corresponding to s.

We define the *terminating probability* of C for reaching a terminal state s' from an initial state s under a scheduler σ as the terminating probability $\mathbb{TP}^\sigma_{(s,C),t_{s'}}$ on M^C. The resulting posterior distribution of C from s under σ is the family $\left(\mathbb{TP}^\sigma_{(s,C),t_{s'}}\right)_{s' \in \mathrm{St}}$ over states.

This posterior distribution forms a subdistribution, i.e.

$$0 \leq \sum_{s' \in \mathrm{St}} \mathbb{TP}^\sigma_{(s,C),t_{s'}} \leq 1,$$

and the sum may be strictly less than 1. In particular, we do not assume almost-sure termination.

2.3 Target Specification and Problem

We conclude this section by formally defining our verification problem. Given a safe region R, the goal is to verify whether the program is always safe; that is, whether every posterior distribution $\left(\mathbb{TP}^\sigma_{(s,C),t_{s'}}\right)_{s' \in \mathrm{St}}$ lies in R for all schedulers σ.

We formally define the safe region R as follows. First, fix a partition $(S_1, \ldots, S_n)$ of St. For a scheduler σ, we define the posterior distribution over the partition $(S_1, \ldots, S_n)$ from an initial state s as

$$\left(P^\sigma_{s,S_i}\right)_{i \in [n]} \quad \text{where} \quad P^\sigma_{s,S_i} \coloneqq \sum_{s_i \in S_i} \mathbb{TP}^\sigma_{(s,C),t_{s_i}}.$$

The safe region R is a non-empty convex set of the form

$$R_\theta \coloneqq \{(p_1, \ldots, p_n) \in [0,1]^n \mid f(p_1, \ldots, p_n) \leq \theta\},$$

where $\theta \in \mathbb{Q}_{\geq 0}$ is a threshold and $f \colon [0,1]^n \to \mathbb{R}_{\geq 0}$ is a monotone polynomial with rational coefficients.

Note that R_θ is downward-closed: if $(p_1, \ldots, p_n) \preceq (p'_1, \ldots, p'_n)$ and $(p'_1, \ldots, p'_n) \in R_\theta$, then $(p_1, \ldots, p_n) \in R_\theta$. This follows from the monotonicity of f.

Our target problem is now formulated as follows:

Algorithm 1 CEGAR Loop

Require: a program C, an initial state s_{init}, a partition $(S_1, \ldots, S_n)$ of St, and a non-empty convex downward closed set $R_\theta \coloneqq \{(p_1, \ldots, p_n) \in [0,1]^n \mid f(p_1, \ldots, p_n) \leq \theta\}$ with a polynomial function f.

1: Initialize h by $[0,1]^n$ that stores results in the iteration.
2: **while** True **do**
3: **if** h implies $\sup_\sigma f(P_1^{C,\sigma}, \ldots, P_n^{C,\sigma}) \leq \theta$ **then**
4: **return** Yes and terminate the loop.
5: **else**
6: Update h based on the found counter example.
7: **end if**
8: **end while**

Problem Statement: Given a program C and an initial state $s \in \mathrm{St}$, the verification problem asks whether, for every scheduler σ, the induced posterior distribution $(P^\sigma_{s,S_1}, \ldots, P^\sigma_{s,S_n})$ belongs to R_θ. Equivalently, the problem reduces to checking whether the following inequality holds:

$$\sup_\sigma f(P^\sigma_{s,S_1}, \ldots, P^\sigma_{s,S_n}) \leq \theta. \tag{1}$$

3 Overview of Our Approach: Exploiting Adequacy and Duality

To present our automated verification method for the target problem, we introduce two key observations that naturally lead to our approach. Alg. 1 provides an overview of the algorithm. There, we maintain an approximation (abstraction) h of C, whose precise details are not needed at this point.

The essence can be summarized as follows: (1) we attempt to prove Eq. (1) using the current approximation h (line 3); and (2) if h is not precise enough to show Eq. (1), we refine h using a counterexample (line 6).

Specifically, the first point is handled by the adequacy of the predicate transformer semantics with respect to the operational semantics [Batz *et al.*, 2024]. The second point is addressed using the Fenchel-Moreau duality from convex optimization (cf. [Boyd and Vandenberghe, 2004]).

3.1 Adequacy

We first address how sufficiently precise abstractions h allow us to conclude that all posterior distributions lie in R_θ. We explain our approach as follows. Assume we have a finite sequence $(\mathbf{w}_1, u_1), \ldots, (\mathbf{w}_m, u_m)$ of pairs consisting of a weight vector $\mathbf{w}_i \in [0,1]^n$ and a bound $u_i \in [0,1]$ such that

$$\sup_\sigma \ \mathbf{w}_i \cdot (P^\sigma_{s,S_1}, \ldots, P^\sigma_{s,S_n}) \leq u_i.$$

We call such a finite sequence a *valid sequence*. The following property then holds:

Lemma 1. *Given a valid sequence* $(\mathbf{w}_1, u_1), \ldots, (\mathbf{w}_m, u_m)$*, if the following property holds:*

$$\bigcap_{i \in [m]} \left\{(p_1, \ldots, p_n) \in [0,1]^n \,\middle|\, \mathbf{w}_i \cdot (p_1, \ldots, p_n) \leq u_i\right\} \subseteq R_\theta. \tag{2}$$

Then Eq. (1) *holds.*

Proof. By the definition of a valid sequence, every posterior distribution $(P^\sigma_{s,S_1}, \dots, P^\sigma_{s,S_n})$ lies in the set $\left\{(p_1, \dots, p_n) \in [0,1]^n \,\middle|\, \mathbf{w}_i \cdot (p_1, \dots, p_n) \leq u_i\right\}$ for each $i \in [m]$. This yields the desired implication. □

Conveniently, Eq. (2) can be checked as follows. Since the set

$$\bigcap_{i \in [m]} \{\mathbf{p} \in [0,1]^n \mid \mathbf{w}_i \cdot \mathbf{p} \leq u_i\}$$

is a convex polytope in $[0,1]^n$, it has finitely many vertices $\mathbf{p}_1, \dots, \mathbf{p}_\ell$ that generate it, i.e. it is the smallest closed convex set containing these vertices (see, e.g., [Boyd and Vandenberghe, 2004]).

As R_θ is also a closed convex set, it suffices to check whether all vertices $\mathbf{p}_1, \dots, \mathbf{p}_\ell$ lie in R_θ, which is straightforward since f is a polynomial, assuming all $\mathbf{w}_i$ and u_i are rational. Such vertices can be extracted by standard algorithms (e.g. via polyhedral libraries including [Jeannet and Miné, 2009]). In fact, in our CEGAR loop the valid sequence does serve as the abstraction h.

Now an important question is how to obtain such a valid sequence. We achieve this by exploiting the adequacy [Batz *et al.*, 2024] of the *predicate transformer semantics* (or *weakest pre-expectation semantics*) [Kozen, 1985; McIver and Morgan, 2005] with respect to the operational semantics.

Formally, for a program C, the predicate transformer semantics yields a function $\Phi^C \colon [0,1]^{\mathrm{St}} \to [0,1]^{\mathrm{St}}$, called the predicate transformer of C, which serves as a denotational semantics. Adequacy characterizes how Φ^C relates to the operational semantics of C.

Lemma 2 (adequacy)**.** *Given a program C and a vector $\mathbf{w} \in [0,1]^n$ (called weight), we have $\Phi^C(\overline{\mathbf{w}})(s) = \sup_\sigma \mathbf{w} \cdot (P^\sigma_{s,S_1}, \dots, P^\sigma_{s,S_n})$, where $\overline{\mathbf{w}} \colon \mathrm{St} \to [0,1]$ is given by $\overline{\mathbf{w}}(s) = \mathbf{w}(i)$ if $s \in S_i$; recall that $(S_1, \dots, S_n)$ is a partition of* St.

Proof. It follows by Theorem 6 in [Batz *et al.*, 2024]. □

Thanks to well-known properties of predicate transformers—such as compositionality and least-fixed-point characterizations (see, e.g., [Batz *et al.*, 2024])—there exist automated verification methods (e.g., [Chatterjee *et al.*, 2017; Takisaka *et al.*, 2021; Batz *et al.*, 2023]) that compute an upper bound u on $\Phi^C(\overline{\mathbf{w}})$, establishing

$$\Phi^C(\overline{\mathbf{w}})(s) = \sup_\sigma \mathbf{w} \cdot (P^\sigma_{s,S_1}, \dots, P^\sigma_{s,S_n}) \leq u.$$

Given a weight vector $\mathbf{w} \in [0,1]^n$, we may be possible to obtain an upper bound u by these existing automated methods, which in turn becomes an element in a valid sequence.

3.2 Fenchel-Moreau Duality

We now explain how to refine the abstraction h, which consists of a valid sequence $(\mathbf{w}_1, u_1), \dots, (\mathbf{w}_m, u_m)$, in the case where h does not imply Eq. (2). Recall that the safe region R is assumed to be a non-empty downward-closed convex subset of the unit cube $[0,1]^n$. Since f is a polynomial, R is also closed by continuity.

From a duality perspective, R admits a representation as the intersection of half-spaces. Formally, we have the following representation theorem:

Proposition 1 (duality)**.** *A safety region $R \in [0,1]^n$ can be rewritten by*

$$R = \bigcap_{\mathbf{w} \in [0,1]^n} \left\{(p_1, \dots, p_n) \in [0,1]^n \,\middle|\, \mathbf{w} \cdot (p_1, \dots, p_n) \leq \sup_{\mathbf{p} \in R} \mathbf{w} \cdot \mathbf{p}\right\}.$$

Proof. This follows by the bijective correspondence of Fenchel conjugates. To see this, get the indicator function $\delta_R \colon [0,1]^n \to [0,\infty]$ of R defined by $\delta_R(\mathbf{p}) \coloneqq 0$ if $\mathbf{p} \in R$ and $\delta_R(\mathbf{p}) \coloneqq \infty$ otherwise. Since δ_R is a proper lower-semi continuous convex function, δ_R has its Fenchel conjugate $(\delta_R)^*$, where the mapping satisfies $(\delta_R)^{**} = \delta_R$ by the Fenchel-Moreau theorem. By calculating $(\delta_R)^{**}$, we have the half-space representation. Note that we only need non-negative weight vectors $\mathbf{w} \in [0,1]^n$ since R is downward closed in $[0,1]^n$. See, e.g., [Boyd and Vandenberghe, 2004] for the detail of the Fenchel conjugate. □

An important consequence of the duality theorem concerns certificates for counterexamples $\mathbf{p} \notin R$, where in our algorithm $\mathbf{p}$ is one of the vertices $\mathbf{p}_1, \dots, \mathbf{p}_\ell$ generating the convex polytope. By Prop. 1, the fact that $\mathbf{p} \notin R$ guarantees the existence of a weight $\mathbf{w} \in [0,1]^n$ such that

$$\mathbf{w} \cdot \mathbf{p} > \sup_{\mathbf{q} \in R} \mathbf{w} \cdot \mathbf{q},$$

which serves as a certificate for $\mathbf{p} \notin R$. Within our CEGAR loop, such a weight $\mathbf{w}$ is automatically synthesised based on the discovered counterexample. Once $\mathbf{w}$ is obtained, we compute an upper bound u with respect to $\mathbf{w}$ such that $u \leq \sup_{\mathbf{q} \in R} \mathbf{w} \cdot \mathbf{q} < \mathbf{w} \cdot \mathbf{p}$ using existing techniques, and refine h by updating

$$h \coloneqq (\mathbf{w}_1, u_1), \dots, (\mathbf{w}_m, u_m), (\mathbf{w}, u).$$

4 Our Algorithm

In this section, we present our algorithm in full detail. We already sketched the overall workflow in Alg. 1; here, we elaborate on the procedure described in lines 3-7 of Alg. 1.

Specifically, Alg. 2 presents the procedure in detail. Recall that $h = (\mathbf{w}_i, u_i)_{i \in I}$ is a valid sequence, and let $\mathrm{CP}(h)$ denote the convex polytope induced by h. To verify whether this valid sequence entails Eq. (1) via Lem. 1, we first compute the vertices $\mathbf{p}_1, \dots, \mathbf{p}_m$ of $\mathrm{CP}(h)$ (line 1). We then initialize variables $\epsilon \in \mathbb{R}_{\geq 0}$ and $\mathbf{r}, \mathbf{w} \in [0,1]^n$, which will encode a counterexample in case h fails to imply Eq. (1).

For each vertex $\mathbf{p} \in \{\mathbf{p}_1, \dots, \mathbf{p}_m\}$, we execute the subroutine in lines 4-9. First, we check whether $\mathbf{p} \in R_\theta$, where

$$R_\theta = \{(p_1, \dots, p_n) \in [0,1]^n \mid f(p_1, \dots, p_n) \leq \theta\},$$

as shown in line 4. Since f is a polynomial, this membership test is straightforward.

If $\mathbf{p} \notin R_\theta$, we estimate how *far* $\mathbf{p}$ is from the safe region R_θ. To this end, we solve the optimisation problem in line

5. Intuitively, we compute the closest point $\mathbf{q} \in R_\theta$ to $\mathbf{p}$, which yields a weight $\mathbf{w}$ separating $\mathbf{p}$ and $\mathbf{q}$, meaning that $\mathbf{w} \cdot \mathbf{q} < \mathbf{w} \cdot \mathbf{p}$. Note that the minimiser $\mathbf{q}$ is unique because the function $\| \cdot \|_2^2$ is strictly convex. Moreover, the following desirable property holds:

Lemma 3. *Assume that* $\mathbf{q}$ *is the minimiser and* $\mathbf{w} = \mathbf{p} - \mathbf{q}$. *We have* $\mathbf{w} \in [0,1]^n$ *and*

$$\mathbf{w} \cdot \mathbf{q} = \max_{\mathbf{x} \in R_\theta} \mathbf{w} \cdot \mathbf{x}.$$

Proof. It is easy to see that $\mathbf{w} \in [0,1]^n$ because R_θ is downward closed. Since $\mathbf{q}$ is the minimiser of $\|\mathbf{q} - \mathbf{p}\|_2$ in R_θ, we have $(\mathbf{p} - \mathbf{q}) \cdot (\mathbf{x} - \mathbf{q}) \leq 0$ for all $\mathbf{x} \in R_\theta$. This implies $(\mathbf{p} - \mathbf{q}) \cdot \mathbf{x} \leq (\mathbf{p} - \mathbf{q}) \cdot \mathbf{q}$, for all $\mathbf{x} \in R_\theta$. Since $q \in R_\theta$, we have the desired equality. □

Crucially, Lem. 3 implies that the weight $\mathbf{w}$ separates R_θ and $\mathbf{p}$, since $\max_{\mathbf{x} \in R_\theta} \mathbf{w} \cdot \mathbf{x} < \mathbf{w} \cdot \mathbf{p}$. It is therefore natural to choose $\mathbf{w} := \mathbf{p} - \mathbf{q}$ and, by Lem. 2, attempt to verify $\Phi^C(\overline{\mathbf{w}})(s) \leq \mathbf{w} \cdot \mathbf{q}$ in order to exclude the point $\mathbf{p}$ from the current abstraction (valid sequence) h.

Since verifying $\Phi^C(\overline{\mathbf{w}})(s) \leq \mathbf{w} \cdot \mathbf{q}$ is often the bottleneck, we only store the minimizer $\mathbf{q}$ and the weight $\mathbf{w}$ of the farthest point $\mathbf{p}$, as handled in lines 6-8.

In line 11, we check whether $\epsilon = 0$, which indicates that all points $\mathbf{p}_1, \ldots, \mathbf{p}_m$ lie in R_θ. Otherwise, we attempt to exclude the farthest point $\mathbf{p} \notin R_\theta$ by finding an upper bound u on $\Phi^C(\overline{\mathbf{w}})(s)$ such that $u \leq \mathbf{w} \cdot \mathbf{q}$.

Whenever such an upper bound is found, our algorithm strictly refines the valid sequence h. We formally state this progress property below:

Theorem 1 (progress)**.** *In line 14, suppose that we find an upper bound* u *that satisfies the condition, then the new valid sequence* $h' := (\mathbf{w}_1, u_1), \ldots, (\mathbf{w}_{|I|}, u_{|I|}), (\mathbf{w}, u)$ *satisfies* $\mathrm{CP}(h') \subsetneq \mathrm{CP}(h)$*, where* h *is the previous valid sequence.*

Proof. Take the farthest point $\mathbf{p}$ of $\mathrm{CP}(h)$ that induces the weight $\mathbf{w}$. Since the upper bound u satisfies $u \leq \mathbf{w} \cdot \mathbf{q} < \mathbf{w} \cdot \mathbf{p}$, it follows that $\mathbf{p} \notin \mathrm{CP}(h')$. Equivalently, the hyperplane $\{\mathbf{x} \mid \mathbf{w} \cdot \mathbf{x} = \mathbf{w} \cdot \mathbf{q}\}$ separates $\mathbf{p}$ from $\mathbf{q}$ (and even from the set R_θ). By construction, we clearly have $\mathrm{CP}(h') \subseteq \mathrm{CP}(h)$. □

Remark 1. *Computing the exact minimizer* $\mathbf{q}$ *may not be possible, as* $\mathbf{q}$ *is not necessarily rational*[1]*. Nevertheless, progress can be ensured if, instead of the minimizer, we find a point* $\mathbf{q} \in R_\theta$ *such that* $\mathbf{q} \preceq \mathbf{p}$ *and obtain an upper bound* u *with respect to the weight* $\mathbf{w} = \mathbf{p} - \mathbf{q}$ *satisfying* $u < \mathbf{w} \cdot \mathbf{p}$*. In our implementation, this condition is enforced as an additional constraint.*

We summarise this section by showing the soundness of our algorithm.

Theorem 2 (soundness)**.** *If Alg. 2 terminates, then Eq.* (1) *holds.*

Proof. It follows by Lem. 1 and the fact that R is a closed convex set. □

[1]We use Z3 [de Moura and Bjørner, 2008] to solve the minimization problem. To the best of our knowledge, it is not clear whether Z3 ensures optimality of the solutions it returns.

Algorithm 2 Refinement by a counter-example

Require: the Requirement in Alg. 1, and $h = (\mathbf{w}_i, u_i)_{i \in I}$.
1: Compute the set $\mathbf{p}_1, \ldots, \mathbf{p}_m$ of vertices of the bounded convex polytope defined by h in $[0,1]^n$.
2: Set a threshold $\epsilon := 0$, and initialize $\mathbf{r} \in [0,1]^n$ and $\mathbf{w} \in [0,1]^n$.
3: **for** $\mathbf{p}$ in $\mathbf{p}_1, \ldots, \mathbf{p}_m$ **do**
4: **if** $\mathbf{p} \notin R_\theta$ **then**
5: Solve the following problem and synthesize the unique solution $\mathbf{q}$:

$$\min_{\mathbf{q} \in R_\theta} \|\mathbf{p} - \mathbf{q}\|_2^2$$

subject to $\mathbf{q} \in R_\theta$ is substochastic ($\sum_i q_i \leq 1$).
6: **if** $\epsilon < \|\mathbf{p} - \mathbf{q}\|_2^2$ **then**
7: Update $\epsilon \leftarrow \|\mathbf{p} - \mathbf{q}\|_2^2$, $\mathbf{r} \leftarrow \mathbf{q}$, and $\mathbf{w} \leftarrow \mathbf{p} - \mathbf{q}$.
8: **end if**
9: **end if**
10: **end for**
11: **if** $\epsilon = 0$ (implies all $\mathbf{p}_1, \ldots, \mathbf{p}_m \in R_\theta$) **then**
12: **return** Yes (Eq. (1) is verified)
13: **else**
14: Compute an upper bound u that satisfies $u \leq \mathbf{w} \cdot \mathbf{r}$ w.r.t. the weight $\mathbf{w}$, and refine h with $(\mathbf{w}, u)$.
15: **end if**

Remark 2 (limitation)**.** *We conclude by noting some limitations of our approach. First, it cannot decide that Eq.* (1) *does not hold. Naively, showing this would require synthesizing a posterior distribution* $(P^\sigma_{s,S_1}, \ldots, P^\sigma_{s,S_n})$ *that violates Eq.* (1)*, which looks challenging and non-trivial; we leave this direction for future work.*

A second limitation concerns the termination of upper-bound computation in line 14*. Our method relies on invariant synthesis via the Knaster-Tarski theorem and, as is common for probabilistic programs over infinite state spaces, termination is not guaranteed. Nevertheless, the procedure is sound: once it terminates, it yields a valid upper bound.*

5 Experimental Evaluation

We evaluate the effectiveness of our approach through preliminary experiments conducted with our prototype implemented in $\mathrm{MuVal}^{\mathrm{QFL}^2}$. Our results demonstrate that the prototype shows promise in addressing the target problem, while also revealing several limitations that present opportunities for future improvement.

5.1 Setting

We implemented our CEGAR loop (Alg. 1), which uses Alg. 2 for refinement, to verify the target problem. To compute the vertices of convex polytopes (line 1 of Alg. 2), we employ Apron [Jeannet and Miné, 2009], and we invoke Z3 [de Moura and Bjørner, 2008] to solve the optimization

[2]Our implementation is available in https://github.com/hiroshi-unno/coar.

Problem	Dim	Deg	Num of Itr	Inv (sec)	OMT (sec)
Intro	2	1	1	1.5	0.0
Intro2	2	2	3	7.4	0.1
F1T10	2	1	1	1.8	0.0
F1T102	2	2	3	9.0	0.1
Choice	2	1	1	1.6	0.0
Choice2	2	2	3	4.8	0.1
Choice3	2	2	3	4.8	0.1
Choice4	2	2	×	×	×
3Dim	3	1	1	3.4	0.0
3Dim2	3	2	4	13.4	0.1
3Dim3	3	2	10	53.7	0.5
3Dim4	3	2	4	13.5	0.2
3Choice	3	2	3	5.8	0.1
3Choice2	3	2	2	3.9	0.1
3Choice3	3	2	2	3.9	0.1
4Dim	4	1	1	39.5	0.0
4Dim2	4	2	×	×	×

Table 1: Experimental results. The symbol × indicates that our solver fails to verity the benchmark, while it satisfies the specification.

problem in line 5 of Alg. 2. In line 14 of Alg. 2, we perform invariant synthesis on top of the existing solver [Chatterjee *et al.*, 2025]. We attempt to synthesise an inductive invariant—i.e., a certificate for an upper bound justified by the Knaster–Tarski theorem—using piecewise polynomial templates. For this, our prototype tries to synthesize a template that admits an inductive invariant by applying Farkas's lemma [Farkas, 1902] and positivity theorems [Handelman, 1988; Putinar, 1993].

For the Case of Linear Constraints. For the safe region R_θ represented by a linear function $f(p_1, \ldots, p_n) = a_1 p_1 + \cdots + a_n p_n$, we employ a simpler procedure instead of Alg. 2. By Lem. 2, the problem can be directly verified if we obtain an upper bound u satisfying $\Phi^C(\overline{\mathbf{w}})(s) \leq u \leq \theta$, where $\mathbf{w} = (a_1, \ldots, a_n)$. Note that all coefficients a_i are non-negative since f must be monotone. Under this condition, our procedure terminates in at most one iteration for the linear constraint.

5.2 Experimental Results

We constructed 17 benchmarks, including the motivating example in §1, to evaluate the performance of our approach. To the best of our knowledge, no existing benchmarks are available for our target problems. We designed our benchmarks using a variety of programs to assess how the size of partitions for posterior distributions—referred to as the *dimension*—and the degree of the polynomial f affect the performance of our implementation.

Following [Kura *et al.*, 2025], each benchmark is written as a fixed-point equation system translated from programs written in pGCL, following the procedure in [Kura *et al.*, 2025]. Roughly speaking, this translation amounts to providing the *characteristic functions* [Batz *et al.*, 2023] of while loops. For each benchmark, we assign a safe region R_θ as a polynomial function f, assuming that R_θ is induced by f satisfying the required assumptions.

The results are summarized in Tab. 1. The experiments were conducted on a machine equipped with an Intel(R) Core(TM) Ultra 5 125U processor and 16 GB of RAM. Dim denotes the dimension of posterior distributions (i.e., the natural number n in $(P_1^\sigma, \ldots, P_n^\sigma)$). Deg denotes the degree of the polynomial defining the safe region; for example, the degree of $f(x, y) = x^2 + y^2$ is 2. Num of Itr indicates the number of CEGAR iterations until termination, Inv denotes the running time of invariant synthesis in seconds, and OMT denotes the running time to solve the optimisation problem (by Z3). The total time for the entire procedure is almost the same as the sum Inv (sec) + OMT (sec), so we omit it from the table. This is because the time required to generate the generators of the convex polytope is very short due to the low dimensionality and the small number of iterations.

5.3 Evaluation

Overall, our CEGAR loop appears to work effectively in low-dimensional settings, particularly in two- and three-dimensional spaces. We now illustrate how our CEGAR loop works on the benchmarks Choice2 and 3Dim2 in detail.

Example 1 (the benchmark Choice2)**.** *The program used in the benchmark is given by*

```
while (x ≥ 3) {
   {C1 } □ {C2}
},
```

where

```
C1 = 1/2 ⋆ (x:=x) + 1/8 ⋆ (x := 0) + 1/4 ⋆
     (x := 1) + 1/8 ⋆ (x := 2),
C2 = 1/2 ⋆ (x:=x) + 1/4 ⋆ (x := 0) + 1/8 ⋆
     (x := 1) + 1/8 ⋆ (x := 2).
```

The safe region R is given by $\{(x, y) \in [0, 1]^2 \mid x^2 + y^2 \leq 9/16\}$, over the two-dimensional posterior distribution $(P^\sigma_{s,[x=1]}, P^\sigma_{s,[x=0]})$. Here $[x = n]$ is the state that assigns the value n to the variable x for each n. From the initial state $x = 4$, our CEGAR loop consructs the following valid sequence $h := (\mathbf{w}_1, u_1), (\mathbf{w}_2, u_2), (\mathbf{w}_3, u_3)$: 1) $\mathbf{w}_1 = (0, 55/144)$ and $u_1 = 55/288$; 2) $\mathbf{w}_2 = (63/128, 0)$ and $u_2 = 63/256$; and 3) $\mathbf{w}_3 = (1/2, 9/16)$ and $u_3 = 127/256$.

Example 2 (the benchmark 3Dim2)**.** *The program used in the benchmark is given by*

```
while (x ≥ 4) {
   {C1 } □ {C2}
},
```

where

```
C1 = 1/2 ⋆ (x:=x) + 1/16 ⋆ (x := 0) + 1/4 ⋆
     (x := 1) + 1/16 ⋆ (x := 2) + 1/8 ⋆ (x :=
     3),
C2 = 1/2 ⋆ (x:=x) + 1/4 ⋆ (x := 0) + 1/16 ⋆
     (x := 1) + 1/16 ⋆ (x := 2) + 1/8 ⋆ (x :=
     3).
```

The safe region R is given by $\{(x, y, z) \in [0, 1]^3 \mid x^2 + y^2 + z^2 \leq 5/8\}$, over the three-dimensional posterior distribution $(P^\sigma_{s,[x=0]}, P^\sigma_{s,[x=1]}, P^\sigma_{s,[x=2]})$. From the initial state $x = 4$,

our CEGAR loop consructs the following valid sequence $h := (\mathbf{w}_1, u_1), \ldots, (\mathbf{w}_4, u_4)$*: 1)* $\mathbf{w}_1 = (0, 137/448, 0)$ *and* $u_1 = 137/896$*; 2)* $\mathbf{w}_2 = (991/3072, 0, 0)$ *and* $u_2 = 991/6144$*; 3)* $\mathbf{w}_3 = (95/3072, 0, 1/2)$ *and* $u_3 = 351/4096$*; and 4)* $\mathbf{w}_4 = (3/4, 49/64, 1/2)$ *and* $u_4 = 2527/4096$.

We observe that tighter specifications tend to require more complex certificates, which are challenging for existing solvers to obtain. In fact, Choice3 and Choice4 share the same underlying program, and because of the tighter specification, our prototype cannot solve Choice4. Lastly, we remark that all benchmarks that require only one iteration have degree 1.

5.4 Limitations

We discuss several limitations of our approach observed through the implementation. The backend solver used in our prototype [Chatterjee *et al.*, 2025; de Moura and Bjørner, 2008] does not support exponential templates, whereas synthesizing inductive invariants to obtain an upper bound u (line 14 of Alg. 2) often requires such templates. Consequently, we only include benchmarks that do not demand highly expressive templates. In our preliminary experiments, our implementation synthesizes only piecewise affine invariants when it successfully computes upper bounds for the benchmarks.

Another limitation concerns scalability with respect to the dimensionality. As shown in Tab. 1, our implementation begins to struggle from dimension 4. One reason is that increasing the number of partitions significantly enlarges the queries generated by [Chatterjee *et al.*, 2025], making satisfiability checking expensive for SMT solvers (we use Z3 [de Moura and Bjørner, 2008]).

Another difficulty arises from the instability of SMT solvers when solving the optimization problem in line 5 of Alg. 2 if the degree is 2. For the benchmark 4Dim2, Z3 fails to solve an optimization problem generated during our CEGAR loop. Notably, by using OptiMathSAT [Sebastiani and Trentin, 2020] instead of Z3, the benchmark 4Dim2 can be solved with two iterations. However, employing OptiMathSAT as the backend SMT solver prevents our implementation from solving most of the other benchmarks; we list the result with OptiMathSAT in Tab. 2.

Problem	Dim	Deg	Num of Itr	Inv (sec)	OMT (sec)
Intro	2	1	1	1.7	0.0
Intro2	2	2	×	×	×
F1T10	2	1	1	2.1	0.0
F1T102	2	2	2	4.4	0.2
Choice	2	1	1	2.0	0.0
Choice2	2	2	×	×	×
Choice3	2	2	×	×	×
Choice4	2	2	×	×	×
3Dim	3	1	1	3.7	0.0
3Dim2	3	2	×	×	×
3Dim3	3	2	×	×	×
3Dim4	3	2	×	×	×
3Choice	3	2	×	×	×
3Choice2	3	2	×	×	×
3Choice3	3	2	×	×	×
4Dim	4	1	1	40.4	0.0
4Dim2	4	2	2	81.4	1.4

Table 2: Experimental results with OptiMathSAT for OMT.

6 Related Work

In this section, we discuss the relationship between our approach and existing work.

6.1 Verifying Posterior Distributions

Posterior distributions of probabilistic programs have been actively studied as a challenging problem. In particular, posterior distributions have been used to reason about equivalence and similarity between programs [Murawski and Ouaknine, 2005]. Such relational reasoning has received considerable attention, and various logical frameworks and sensitivity analyses have been developed in this context [Culpepper and Cobb, 2017; Albarghouthi and Hsu, 2018; Barthe *et al.*, 2016; Huang *et al.*, 2018].

Notably, the recent work [Chatterjee *et al.*, 2024] introduces an automated method for relational reasoning over probabilistic programs with infinite state spaces. Their framework and ours are largely orthogonal, as they address different verification problems. We remark that their framework does not support nondeterminism in probabilistic programs and relies on technical assumptions such as almost-sure termination, whereas our approach supports nondeterminism and does not require almost-sure termination.

6.2 Multi-objective Optimisation

For finite-state MDPs, computing or approximating the set of posterior distributions induced by schedulers has been well studied. Etessami et al. [Etessami *et al.*, 2008] propose multi-objective model-checking algorithms based on approximating the Pareto curve of an MDP. For the multi-objective reachability problem, each point on the Pareto curve corresponds to a Pareto-optimal posterior distribution under a (possibly stochastic) scheduler. Their approach employs the approximation algorithm of Papadimitriou and Yannakakis [Papadimitriou and Yannakakis, 2000], and Forejt et al. [Forejt *et al.*, 2012] later propose an iterative approximation algorithm based on the so-called sandwich algorithm [Rennen *et al.*, 2011; Solanki *et al.*, 1993].

These techniques do not apply to probabilistic programs with infinite state spaces, although our CEGAR loop is inspired by the sandwich algorithm. In particular, the idea of selecting a "good" weight vector $\mathbf{w}$ to improve the approximation is central to the sandwich algorithm. In this paper, we develop a novel CEGAR loop for probabilistic programs that synthesizes such a weight $\mathbf{w}$ based on violations of the current abstraction with respect to a specification (given as a safe region).

6.3 Healthiness and Duality

Our framework is by no means the first to focus on Fenchel-Moreau duality in the context of probabilistic programs. Indeed, Morgan et al. [Morgan *et al.*, 1996] and McIver and Morgan [McIver and Morgan, 2005] already observed this

duality in their study of healthiness conditions for predicate transformers. Such duality aspects in probabilistic programs are well-studied in studies of denotational semantics of higher-order probabilistic programs [Goubault-Larrecq, 2007; Keimel and Plotkin, 2009]

This duality for predicate transformers has further been generalized categorically [Hino *et al.*, 2016], encompassing predicate transformers for finite-state probabilistic programs. Our contribution is to build on this dual viewpoint to develop a novel automated verification method.

7 Conclusion

We formally define a safety problem for posterior distributions of probabilistic programs. To verify such properties automatically, we introduce a novel CEGAR loop that iteratively refines abstractions represented as convex polytopes. We assess the effectiveness of our approach through a preliminary experiment.

As noted in Rem. 2, it would be desirable to automatically synthesize a posterior distribution that serves as a counterexample to Eq. (1). We leave this seemingly challenging problem for future work.

A bottleneck of our approach is the lack of support for exponential templates, which are often needed to compute upper bounds on $\Phi^C(\overline{\mathbf{w}})$. The backend solver we rely on [Chatterjee *et al.*, 2025] currently supports only polynomial templates. A backend capable of handling exponential templates would be highly desirable, potentially by leveraging LLMs to synthesize such invariants; we leave this as future work.

While we focus on proving the safety of probabilistic programs, it would also be useful to be able to disprove the safety of probabilistic programs. This seems to be a more challenging and non-trivial problem, but it may be possible to provide an efficient sound procedure by restricting the language, for example by disallowing non-determinism. We leave this as future work.

Lastly, we aim to improve the scalability of our approach to higher-dimensional problems. In addition to the SMT query size issue observed in §5, several challenges arise in higher dimensions. For example, approximating closed convex sets using half-spaces generally becomes more difficult as the dimensionality increases—a phenomenon also reported in the literature (e.g., [Watanabe *et al.*, 2024]) for finite-state MDPs.

Acknowledgements

We thank the anonymous reviewers for their helpful comments. This study was supported by JSPS KAKENHI Grant Number JP25K24739. K.W. was supported by the JST grant No. JPMJPR25KD.

References

[Albarghouthi and Hsu, 2018] Aws Albarghouthi and Justin Hsu. Synthesizing coupling proofs of differential privacy. *Proc. ACM Program. Lang.*, 2(POPL):58:1–58:30, 2018.

[Ball and Rajamani, 2002] Thomas Ball and Sriram K. Rajamani. The SLAM project: debugging system software via static analysis. In *POPL*, pages 1–3. ACM, 2002.

[Bao *et al.*, 2025] Jialu Bao, Nitesh Trivedi, Drashti Pathak, Justin Hsu, and Subhajit Roy. Data-driven invariant learning for probabilistic programs. *Formal Methods Syst. Des.*, 66(2):278–306, 2025.

[Barthe *et al.*, 2016] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *LICS*, pages 749–758. ACM, 2016.

[Barthe *et al.*, 2020] Gilles Barthe, Joost-Pieter Katoen, and Alexandra Silva, editors. *Foundations of Probabilistic Programming*. Cambridge University Press, 2020.

[Batz *et al.*, 2021] Kevin Batz, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Relatively complete verification of probabilistic programs: an expressive language for expectation-based reasoning. *Proc. ACM Program. Lang.*, 5(POPL):1–30, 2021.

[Batz *et al.*, 2023] Kevin Batz, Mingshuai Chen, Sebastian Junges, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Christoph Matheja. Probabilistic program verification via inductive synthesis of inductive invariants. In *TACAS (2)*, volume 13994 of *LNCS*, pages 410–429. Springer, 2023.

[Batz *et al.*, 2024] Kevin Batz, Benjamin Lucien Kaminski, Christoph Matheja, and Tobias Winkler. J-P: MDP. FP. PP - characterizing total expected rewards in markov decision processes as least fixed points with an application to operational semantics of probabilistic programs. In *Principles of Verification (1)*, volume 15260 of *LNCS*, pages 255–302. Springer, 2024.

[Bingham *et al.*, 2019] Eli Bingham, Jonathan P. Chen, Martin Jankowiak, Fritz Obermeyer, Neeraj Pradhan, Theofanis Karaletsos, Rohit Singh, Paul A. Szerlip, Paul Horsfall, and Noah D. Goodman. Pyro: Deep universal probabilistic programming. *J. Mach. Learn. Res.*, 20:28:1–28:6, 2019.

[Boyd and Vandenberghe, 2004] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[Chakarov and Sankaranarayanan, 2013] Aleksandar Chakarov and Sriram Sankaranarayanan. Probabilistic program analysis with martingales. In *CAV*, volume 8044 of *LNCS*, pages 511–526. Springer, 2013.

[Chatterjee *et al.*, 2017] Krishnendu Chatterjee, Petr Novotný, and Dorde Zikelic. Stochastic invariants for probabilistic termination. In *POPL*, pages 145–160. ACM, 2017.

[Chatterjee *et al.*, 2020] Krishnendu Chatterjee, Hongfei Fu, and Petr Novotný. Termination analysis of probabilistic programs with martingales. In *Foundations of Probabilistic Programming*, pages 221–258. Cambridge University Press, 2020.

[Chatterjee *et al.*, 2024] Krishnendu Chatterjee, Ehsan Kafshdar Goharshady, Petr Novotný, and Dorde Zikelic. Equivalence and similarity refutation for probabilistic programs. *Proc. ACM Program. Lang.*, 8(PLDI):2098–2122, 2024.

[Chatterjee *et al.*, 2025] Krishnendu Chatterjee, Amir Kafshdar Goharshady, Ehsan Kafshdar Goharshady, Mehrdad Karrabi, Milad Saadat, Maximilian Seeliger, and Dorde Zikelic. PolyQEnt: A polynomial quantified entailment solver. In *ATVA*, LNCS, pages 411–424. Springer, 2025.

[Clarke *et al.*, 2003] Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Helmut Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[Culpepper and Cobb, 2017] Ryan Culpepper and Andrew Cobb. Contextual equivalence for probabilistic programs with continuous random variables and scoring. In *ESOP*, volume 10201 of *LNCS*, pages 368–392. Springer, 2017.

[de Moura and Bjørner, 2008] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In *TACAS*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008.

[Etessami *et al.*, 2008] Kousha Etessami, Marta Z. Kwiatkowska, Moshe Y. Vardi, and Mihalis Yannakakis. Multi-objective model checking of Markov decision processes. *Log. Methods Comput. Sci.*, 4(4), 2008.

[Farkas, 1902] Julius Farkas. Theorie der einfachen ungleichungen. *Journal für die reine und angewandte Mathematik (Crelles Journal)*, 1902(124):1–27, 1902.

[Forejt *et al.*, 2012] Vojtech Forejt, Marta Z. Kwiatkowska, and David Parker. Pareto curves for probabilistic model checking. In *ATVA*, volume 7561 of *LNCS*, pages 317–332. Springer, 2012.

[Goodman and Stuhlmüller, 2014] Noah D Goodman and Andreas Stuhlmüller. The Design and Implementation of Probabilistic Programming Languages. http://dippl.org, 2014. Accessed: 2026-1-18.

[Goodman *et al.*, 2008] Noah D. Goodman, Vikash K. Mansinghka, Daniel M. Roy, Kallista A. Bonawitz, and Joshua B. Tenenbaum. Church: a language for generative models. In *UAI*, pages 220–229. AUAI Press, 2008.

[Gordon *et al.*, 2014] Andrew D. Gordon, Thomas A. Henzinger, Aditya V. Nori, and Sriram K. Rajamani. Probabilistic programming. In *FOSE*, pages 167–181. ACM, 2014.

[Goubault-Larrecq, 2007] Jean Goubault-Larrecq. Continuous previsions. In *CSL*, volume 4646 of *LNCS*, pages 542–557. Springer, 2007.

[Gretz *et al.*, 2014] Friedrich Gretz, Joost-Pieter Katoen, and Annabelle McIver. Operational versus weakest pre-expectation semantics for the probabilistic guarded command language. *Perform. Evaluation*, 73:110–132, 2014.

[Handelman, 1988] David Handelman. Representing polynomials by positive linear functions on compact convex polyhedra. *Pacific Journal of Mathematics*, 132(1):35–62, 1988.

[Hino *et al.*, 2016] Wataru Hino, Hiroki Kobayashi, Ichiro Hasuo, and Bart Jacobs. Healthiness from duality. In *LICS*, pages 682–691. ACM, 2016.

[Huang *et al.*, 2018] Zixin Huang, Zhenbang Wang, and Sasa Misailovic. Psense: Automatic sensitivity analysis for probabilistic programs. In *ATVA*, volume 11138 of *LNCS*, pages 387–403. Springer, 2018.

[Jeannet and Miné, 2009] Bertrand Jeannet and Antoine Miné. Apron: A library of numerical abstract domains for static analysis. In *CAV*, volume 5643 of *LNCS*, pages 661–667. Springer, 2009.

[Kaminski and Katoen, 2015] Benjamin Lucien Kaminski and Joost-Pieter Katoen. On the hardness of almost-sure termination. In *MFCS (1)*, volume 9234 of *LNCS*, pages 307–318. Springer, 2015.

[Keimel and Plotkin, 2009] Klaus Keimel and Gordon D. Plotkin. Predicate transformers for extended probability and non-determinism. *Math. Struct. Comput. Sci.*, 19(3):501–539, 2009.

[Kozen, 1985] Dexter Kozen. A probabilistic PDL. *J. Comput. Syst. Sci.*, 30(2):162–178, 1985.

[Kura *et al.*, 2025] Satoshi Kura, Hiroshi Unno, and Takeshi Tsukada. Supermartingales for unique fixed points: A unified approach to lower bound verification. *CoRR*, abs/2504.04132, 2025. To Appear in PLDI2026.

[Li *et al.*, 2025] Guanyan Li, Juanen Li, Zhilei Han, Peixin Wang, Hongfei Fu, and Fei He. Structural abstraction and refinement for probabilistic programs. *Proc. ACM Program. Lang.*, 9(OOPSLA2):1809–1836, 2025.

[McIver and Morgan, 2005] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2005.

[Morgan *et al.*, 1996] Carroll Morgan, Annabelle McIver, and Karen Seidel. Probabilistic predicate transformers. *ACM Trans. Program. Lang. Syst.*, 18(3):325–353, 1996.

[Murawski and Ouaknine, 2005] Andrzej S. Murawski and Joël Ouaknine. On probabilistic program equivalence and refinement. In *CONCUR*, volume 3653 of *LNCS*, pages 156–170. Springer, 2005.

[Ngo *et al.*, 2018] Van Chan Ngo, Quentin Carbonneaux, and Jan Hoffmann. Bounded expectations: resource analysis for probabilistic programs. In *PLDI*, pages 496–512. ACM, 2018.

[Papadimitriou and Yannakakis, 2000] Christos H. Papadimitriou and Mihalis Yannakakis. On the approximability of trade-offs and optimal access of web sources. In *FOCS*, pages 86–92. IEEE Computer Society, 2000.

[Plotkin, 2004] Gordon D. Plotkin. The origins of structural operational semantics. *J. Log. Algebraic Methods Program.*, 60-61:3–15, 2004.

[Puterman, 1994] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[Putinar, 1993] Mihai Putinar. Positive polynomials on compact semi-algebraic sets. *Indiana University Mathematics Journal*, 42(3):969–984, 1993.

[Rennen *et al.*, 2011] Gijs Rennen, Edwin R. van Dam, and Dick den Hertog. Enhancement of sandwich algorithms for approximating higher-dimensional convex Pareto sets. *INFORMS J. Comput.*, 23(4):493–517, 2011.

[Ritchie *et al.*, 2015] Daniel Ritchie, Ben Mildenhall, Noah D. Goodman, and Pat Hanrahan. Controlling procedural modeling programs with stochastically-ordered sequential Monte Carlo. *ACM Trans. Graph.*, 34(4):105:1–105:11, 2015.

[Sebastiani and Trentin, 2020] Roberto Sebastiani and Patrick Trentin. Optimathsat: A tool for optimization modulo theories. *J. Autom. Reason.*, 64(3):423–460, 2020.

[Solanki *et al.*, 1993] Rajendra S Solanki, Perry A Appino, and Jared L Cohon. Approximating the noninferior set in multiobjective linear programming problems. *European journal of operational research*, 68(3):356–373, 1993.

[Takisaka *et al.*, 2021] Toru Takisaka, Yuichiro Oyabu, Natsuki Urabe, and Ichiro Hasuo. Ranking and repulsing supermartingales for reachability in randomized programs. *ACM Trans. Program. Lang. Syst.*, 43(2):5:1–5:46, 2021.

[Tolpin *et al.*, 2016] David Tolpin, Jan-Willem van de Meent, Hongseok Yang, and Frank D. Wood. Design and implementation of probabilistic programming language Anglican. In *IFL*, pages 6:1–6:12. ACM, 2016.

[van de Meent *et al.*, 2018] Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, and Frank Wood. An introduction to probabilistic programming. *CoRR*, abs/1809.10756, 2018.

[Watanabe *et al.*, 2024] Kazuki Watanabe, Marck van der Vegt, Ichiro Hasuo, Jurriaan Rot, and Sebastian Junges. Pareto curves for compositionally model checking string diagrams of MDPs. In *TACAS (2)*, volume 14571 of *LNCS*, pages 279–298. Springer, 2024.

[Wood *et al.*, 2014] Frank D. Wood, Jan-Willem van de Meent, and Vikash Mansinghka. A new approach to probabilistic programming inference. In *AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 1024–1032. JMLR.org, 2014.