MDPs as Distribution Transformers: Affine Invariant Synthesis for Safety Objectives

S. Akshay[1(✉)], Krishnendu Chatterjee[2], Tobias Meggendorfer[2,3], and Đorđe Žikelić[2]

[1] Indian Institute of Technology Bombay, Mumbai, India
akshayss@cse.iitb.ac.in
[2] Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
{krishnendu.chatterjee,dzikelic}@ist.ac.at
[3] Technical University of Munich, Munich, Germany
tobias.meggendorfer@cit.tum.de

Abstract. Markov decision processes can be viewed as transformers of probability distributions. While this view is useful from a practical standpoint to reason about trajectories of distributions, basic reachability and safety problems are known to be computationally intractable (i.e., Skolem-hard) to solve in such models. Further, we show that even for simple examples of MDPs, strategies for safety objectives over distributions can require infinite memory and randomization.

In light of this, we present a novel overapproximation approach to synthesize strategies in an MDP, such that a safety objective over the distributions is met. More precisely, we develop a new framework for template-based synthesis of certificates as affine distributional and inductive invariants for safety objectives in MDPs. We provide two algorithms within this framework. One can only synthesize memoryless strategies, but has relative completeness guarantees, while the other can synthesize general strategies. The runtime complexity of both algorithms is in PSPACE. We implement these algorithms and show that they can solve several non-trivial examples.

Keywords: Markov decision processes · invariant synthesis · distribution transformers · Skolem hardness

1 Introduction

Markov decision processes (MDPs) are a classical model for probabilistic decision making systems. They extend the basic probabilistic model of Markov chains with non-determinism and are widely used across different domains and contexts. In the

This work was supported in part by the ERC CoG 863818 (FoRM-SMArt) and the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 665385 as well as DST/CEFIPRA/INRIA project EQuaVE and SERB Matrices grant MTR/2018/00074.

C. Enea and A. Lal (Eds.): CAV 2023, LNCS 13966, pp. 86–112, 2023.
https://doi.org/10.1007/978-3-031-37709-9_5

verification community, MDPs are often viewed through an automata-theoretic lens, as state transformers, with runs being sequences of states with certain probability for taking each run (see e.g., [9]). With this view, reachability probabilities can be computed using simple fixed point equations and model checking can be done over appropriately defined logics such as PCTL*. However, in several contexts such as modelling biochemical networks, queueing theory or probabilistic dynamical systems, it is more convenient to view MDPs as transformers of probability distributions over the states, and define objectives over these distributions [1,5,12,17,44,47]. In this framework, we can, for instance, easily reason about properties such as the probability in a set of states always being above a given threshold or comparing the probability in two states at some future time point. More concretely, in a chemical reaction network, we may require that the concentration of a particular complex is never above 10%. Such distribution-based properties cannot be expressed in PCTL* [12], and thus several orthogonal logics have been defined [1,12,44] that reason about distributions.

Unfortunately, and perhaps surprisingly, when we view them as distribution transformers even the simplest reachability and safety problems with respect to probability distributions over states remain unsolved. The reason for this is a number-theoretical hardness result that lies at the core of these questions. In [3], it is shown that even with just Markov chains, reachability is as hard as the so-called Skolem problem, and safety is as hard as the Positivity problem [55, 56], the decidability of both of which are long-standing open problems in linear recurrence sequences. Moreover, synthesizing strategies that resolve the non-determinism in MDPs to achieve an objective (whether reachability or safety) is further complicated by the issue of how much memory can be allowed for the strategy. As we show in Sect. 3, even for very simple examples, strategies for safety can require infinite memory as well as randomization.

In light of these difficulties, what can one do to tackle these problems *in theory and in practice*? In this paper, we take an over-approximation route to approach these questions, not only to check existence of strategies for safety but also synthesize them. Inspired by the success of invariant synthesis in program verification, our goal is to develop a novel invariant-synthesis based approach towards strategy synthesis in MDPs, viewed as transformers of distributions. In this paper, we restrict our attention to a class of safety objectives on MDPs, which are already general enough to capture several interesting and natural problems on MDPs. Our contributions are the following:

1. We define the notion of *inductive distributional invariants* for safety in MDPs. These are sets of probability distributions over states of the MDP, that (i) contain all possible distributions reachable from the initial distribution, under all strategies of an MDP, and (ii) are closed under taking the next step.
2. We show that such invariants provide *sound and complete certificates* for proving safety objectives in MDPs. In doing so, we formalize the link between strategies and distributional invariants in MDPs. This by itself does not help us get effective algorithms in light of the hardness results above. Hence we then focus on synthesizing invariants of a particular *shape*.

3. We develop two algorithms for automated synthesis of *affine* inductive distributional invariants that prove safety in MDPs, and *at the same time*, synthesize the associated strategies.
   - The first algorithm is restricted to synthesizing memoryless strategies but is *relatively complete*, i.e., whenever a memoryless strategy and an affine inductive distributional invariant that witness safety exist, we are guaranteed to find them.
   - The second algorithm can synthesize general strategies as well as memoryless strategies, but is incomplete in general.

   In both cases, we employ a template-based synthesis approach and reduce synthesis to the existential first-order theory of reals, which gives a PSPACE complexity upper bound. In the first case, this reduction depends on Farkas' lemma. In the second case, we need to use Handelman's theorem, a specialized result for strictly positive polynomials.
4. We implement our approaches and show that for several practical and non-trivial examples, affine invariants suffice. Further, we demonstrate that our prototype tool can synthesize these invariants and associated strategies.

Finally, we discuss the generalization of our approach from affine to polynomial invariants and some variants that our approach can handle.

1.1 Related Work

Distribution-based Safety Analysis in MDPs. The problem of checking distribution-based safety objectives for MDPs was defined in [5] but a solution was provided only in the *uninitialized* setting, where the initial distribution is not given and also under the assumption that the target set is closed and bounded. In contrast, we tackle both initialized and uninitialized settings, our target sets are general affine sets and we focus on actually synthesizing strategies not just proving existence.

Template-based Program Analysis. Template-based synthesis via the means of linear/polynomial constraint solving is a standard approach in program analysis to synthesizing certificates for proving properties of programs. Many of these methods utilize Farkas' lemma or Handelman's theorem to automate the synthesis of program invariants [20,27], termination proofs [6,14,23,28,57], reachability proofs [8] or cost bounds [16,39,64]. The works [2,18,19,21,22,24,25,62,63] utilize Farkas' lemma or Handelman's theorem to synthesize certificates for these properties in probabilistic programs. While our algorithms build on the ideas from the works on template-based inductive invariant synthesis in programs [20,27], the key novelty of our algorithms is that they synthesize a fundamentally different kind of invariants, i.e. *distributional invariants* in MDPs. In contrast, the existing works on (probabilistic) program analysis synthesize *state* invariants. Furthermore, our algorithms synthesize distributional invariants *together* with MDP strategies. While it is common in controller synthesis

to synthesize an MDP strategy for a *state* invariant, we are not aware of any previous work that uses template-based synthesis methods to compute MDP strategies for a *distributional* invariant.

Other Approaches to Invariant Synthesis in Programs. Alternative approaches to invariant synthesis in programs have also been considered, for instance via abstract interpretation [29,30,33,60], counterexample guided invariant synthesis (CEGIS) [7,10,34], recurrence analysis [32,42,43] or learning [35,61]. While some of these approaches can be more scalable than constraint solving-based methods, they typically do not provide relative completeness guarantees. An interesting direction of future work would be to explore whether these alternative approaches could be used for synthesizing distributional invariants together with MDP strategies more efficiently.

Weakest Pre-expectation Calculus. Expectation transformers and the weakest pre-expectation calculus generalize Dijkstra's weakest precondition calculus to the setting of probabilistic programs. Expectation transformers were introduced in the seminal work on probabilistic propositional dynamic logic (PPDL) [45] and were extended to the setting of probabilistic programs with non-determinism in [48,52]. Weakest pre-expectation calculus for reasoning about expected runtime of probabilistic programs was presented in [40]. Intuitively, given a function over probabilistic program outputs, the weakest pre-expectation calculus can be used to reason about the supremum or the infimum expected value of the function upon executing the probabilistic program, where the supremum and the infimum are taken over the set of all possible schedulers (i.e. strategies) used to resolve non-determinism. When the function is the indicator function of some output set of states, this yields the method for reasoning about the probability of reaching the set of states. Thus, weakest pre-expectation calculus allows reasoning about safety with respect to *sets of states*. In contrast, we are interested in reasoning about safety with respect to *sets of probability distribution over states*. Moreover, while the expressiveness of this calculus allows reasoning about very complex programs, its automation typically requires user input. In this work, we aim for a fully automated approach to checking distribution-based safety.

2 Preliminaries

In this section, we recall basics of probabilistic systems and set up our notation. We assume familiarity with the central ideas of measure and probability theory, see [13] for a comprehensive overview. We write $[n] := \{1, \ldots, n\}$ to denote the set of all natural numbers from 1 to n. For any set S, we use $\overline{S}$ to denote its complement. A *probability distribution* on a countable set X is a mapping $\mu : X \to [0,1]$, such that $\sum_{x \in X} \mu(x) = 1$. Its *support* is denoted by $\mathrm{supp}(\mu) = \{x \in X \mid \mu(x) > 0\}$. We write $\Delta(X)$ to denote the set of all probability distributions on X. An event happens *almost surely* (a.s.) if it happens with probability 1. We assume that countable sets of states S are equipped with an arbitrary but fixed numbering.

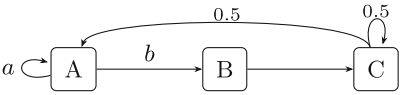

Fig. 1. Our running example MDP. It comprises three states $S = \{A, B, C\}$, depicted by rounded rectangles. In state A, there are two actions available, namely a and b. We have $\delta(A, a, A) = 1$ and $\delta(A, b, B) = 1$, indicated by arrows. States B and C have only one available action each, thus we omit explicitly labelling them.

2.1 Markov Systems

A *(discrete time) Markov chain (MC)* is a tuple $\mathsf{M} = (S, \delta)$, where S is a finite set of *states* and $\delta : S \to \Delta(S)$ a *transition function*, assigning to each state a probability distribution over successor states. A *Markov decision process (MDP)* is a tuple $\mathcal{M} = (S, Act, \delta)$, where S is a finite set of *states*, Act is a finite set of *actions*, overloaded to yield for each state s the set of *available actions* $Act(s) \subseteq Act$, and $\delta : S \times Act \to \Delta(S)$ is a *transition function* that for each state s and (available) action $a \in Act(s)$ yields a probability distribution over successor states. For readability, we write $\delta(s, s')$ and $\delta(s, a, s')$ instead of $\delta(s)(s')$ and $\delta(s, a)(s')$, respectively. By abuse of notation, we redefine $S \times Act := \{(s, a) \mid s \in S \wedge a \in Act(s)\}$ to refer to the set of state-action pairs. See Fig. 1 for an example MDP. This MDP is our running example and we refer to it throughout this work to point out some of the peculiarities.

An *infinite path* in an MC is an infinite sequence $\rho = s_1 s_2 \cdots \in S^\omega$, such that for every $i \in \mathbb{N}$ we have $\delta(s_i, s_{i+1}) > 0$. A *finite path* ϱ is a finite prefix of an infinite path. Analogously, infinite paths in MDP are infinite sequences $\rho = s_1 a_1 s_2 a_2 \cdots \in (S \times Act)^\omega$ such that $a_i \in Act(s_i)$ and $\delta(s_i, a_i, s_{i+1}) > 0$ for every $i \in \mathbb{N}$, and finite paths are finite prefixes thereof. We use ρ_i and ϱ_i to refer to the i-th state in the given (in)finite path, and IPaths_M and FPaths_M for the set of all (in)finite paths of a system M.

Semantics. A Markov chain evolves by repeatedly applying the probabilistic transition function in each step. For example, if we start in state s_1, we obtain the next state s_2 by drawing a random state according to the probability distribution $\delta(s_1)$. Repeating this ad infinitum produces a random infinite path. Indeed, together with an initial state s, a Markov chain M induces a unique probability measure $\mathsf{Pr}_{\mathsf{M},s}$ over the (uncountable) set of infinite paths [9].

This reasoning can be lifted to distributions over states, as follows. Suppose we begin in $\mu_0 = \{s_1 \mapsto 0.5, s_2 \mapsto 0.5\}$, meaning that initially we are in state s_1 or s_2 with probability 0.5 each. Then, $\mu_1(s') = \mu_0(s_1) \cdot \delta(s_1, s') + \mu_0(s_2) \cdot \delta(s_2, s')$, i.e. the probability to be in a state s' in the next step is 0.5 times the probability of moving from s_1 and s_2 there, respectively. For an initial distribution, we likewise obtain a probability distribution over infinite paths by setting $\mathsf{Pr}_{\mathsf{M},\mu_0}[S] := \sum_{s \in S} \mu_0(s) \cdot \mathsf{Pr}_{\mathsf{M},s}[S]$ for measurable $S \subseteq \mathsf{IPaths}_{\mathsf{M}}$.

In contrast to Markov chains, MDPs also feature non-determinism, which needs be resolved in order to obtain probabilistic behaviour. This is achieved

by *(path) strategies*, recipes to resolve non-determinism. Formally, a strategy on an MDP classically is defined as a function $\pi : \mathsf{FPaths}_{\mathcal{M}} \to \Delta(Act)$, which given a finite path $\varrho = s_0 a_0 s_1 a_1 \dots s_n$ yields a probability distribution $\pi(\varrho) \in \Delta(Act(s_n))$ on the actions to be taken next. We write Π to denote the set of all strategies. Fixing any strategy π induces a Markov chain $\mathcal{M}^\pi = (\mathsf{FPaths}_{\mathcal{M}}, \delta^\pi)$, where for a state $\varrho = s_0 a_0 \dots s_n \in \mathsf{FPaths}_{\mathcal{M}}$ the successor distribution is defined as $\delta^\pi(\varrho, \varrho a_{n+1} s_{n+1}) = \pi(\varrho, a_{n+1}) \cdot \delta(s_n, a_{n+1}, s_{n+1})$. (Note that the state space of this Markov chain in general is countably infinite.) Consequently, for each strategy π and initial distribution μ_0 we also obtain a unique probability measure $\mathsf{Pr}_{\mathcal{M}^\pi, \mu_0}$ on the infinite paths of $\mathcal{M}$. (Technically, the MC $\mathcal{M}^\pi$ induces a probability measure over paths in $\mathcal{M}^\pi$, i.e. paths where each element is a finite path of $\mathcal{M}$, however this can be directly projected to a measure over $\mathsf{IPaths}_{\mathcal{M}}$.)

A *one-step strategy* (also known as *memoryless* or *positional* strategy) corresponds to a fixed choice in each state, independent of the history, i.e. a mapping $\pi : S \to \Delta(Act)$. Fixing such a strategy induces a finite state Markov chain $\mathcal{M}^\pi = (S, \delta^\pi)$, where $\delta^\pi(s, s') = \sum_{a \in Act(s)} \pi(s)(a) \cdot \delta(s, a, s')$. We write Π_1 for the set of all one-step strategies.

A sequence of one-step strategies $(\pi_i) \in \Pi_1^\omega$ induces a general strategy which in each step i and state s chooses $\pi_i(s)$. Observe that aside from the state, such a strategy only depends on the current step, also called *Markov strategy*.

2.2 MDPs as Distribution Transformers

Probabilistic systems typically are viewed as "random generators" for paths, and we consequently investigate the (expected) behaviour of a generated path, i.e. path properties. However, in this work we follow a different view, and treat systems as *transformers of distributions*. Formally, fix a Markov chain M. For a given initial distribution μ_0, we can define the distribution at step i by $\mu_i(s) = \mathsf{Pr}_{\mu_0}[\{\rho \in \mathsf{IPaths}_{\mathsf{M}} \mid \rho_i = s\}]$. We write $\mu_i = \mathsf{M}(\mu_0, i)$ for the i-th distribution and $\mu_1 = \mathsf{M}(\mu_0)$ for the "one-step" application of this transformation. Likewise, we obtain the same notion for an MDP $\mathcal{M}$ combined with a strategy π, and write $\mu_i = \mathcal{M}^\pi(\mu_0, i)$, $\mu_1 = \mathcal{M}^\pi(\mu_0)$. In summary, for a given initial distribution, a Markov chain induces a unique stream of distributions, and an MDP provides one for each strategy.

This naturally invites questions related to this induced stream of distributions. In their path interpretation, queries such as *reachability* or *safety*, i.e. asking the probability of reaching or avoiding a set of states, allow for simple, polynomial time solutions [9,58]. However, the corresponding notions already are surprisingly difficult in the space of distributions. Thus, we restrict to the *safety problem*, which we introduce in the following. Intuitively, given a *safe set* of distributions over states $H \subseteq \Delta(S)$, we are interested in deciding whether the MDP can be controlled such that the stream of distributions always remains inside H.

3 Problem Statement and Examples

Let $\mathcal{M} = (S, Act, \delta)$ be an MDP and $H \subseteq \Delta(S)$ be a safe set. A distribution μ_0 is called *H-safe under π* if $\mathcal{M}^\pi(\mu_0, i) \in H$ for all $i \geq 0$, and *H-safe* if there exists a strategy under which μ_0 is safe. We mention two variants of the resulting decision problem as defined in [5]:

- Initialized safety: Given an initial probability distribution μ_0 and safe set H, decide whether μ_0 is H-safe.
- Uninitialized safety: Given a safe set H, decide whether there exists a distribution μ which is H-safe.

Note that we have discussed neither the shape nor the representation of H, which naturally plays an important role for decidability and complexity.

One may be tempted to think that the initialized variant is simpler, as more input is given. However, this problem is known to be Positivity-*hard*[1] already for simple cases and already when H is defined in terms of rational constants!

Theorem 1 ([3]). *The initialized safety problem for Markov chains and H given as linear inequality constraint ($H = \{\mu \mid \mu(s) \leq r, s \in S, r \in \mathbb{Q} \cap [0,1]\}$), is* Positivity-*hard.*

Proof. In [3, Corollary 4], the authors show that the inequality version of the Markov reachability problem, i.e. deciding whether there exists an i such that $\mu_i(s) > r$ for a given rational r, is Positivity-hard. The result follows by observing that safety is the negation of reachability. □

Thus, finding a decision procedure for this problem is unlikely, since it would answer several fundamental questions of number theory, see e.g. [41,55,56]. In contrast, the uninitialized problem is known to be decidable for safe sets H given as closed, convex polytopes (see [5] for details and [1] for a different approach specific to Markov chains). In a nutshell, we can restrict to the potential fixpoints of $\mathcal{M}$, i.e. all distributions μ such that $\mu = \mathcal{M}^\pi(\mu, i)$ for some strategy π. It turns out that this set of distributions is a polytope and the problem – glossing over subtleties – reduces to checking whether the intersection of H with this polytope is non-empty. However, we note that the solution of [5] does not yield the witness strategy. In the following, we thus primarily focus on the initialized question. In Sect. 6, we then show how our approach, which also synthesizes a witness strategy, is directly applicable to the uninitialized case.

In light of the daunting hardness results for the general initialized problem, we restrict to *affine linear safe sets*, i.e. H which are specified by a finite set of affine linear inequalities. Formally, these sets are of the form $H = \{\mu \in \Delta(S) \mid \bigwedge_{j=1}^{N}(c_0^j + \sum_{i=1}^{n} c_i^j \cdot \mu(s_i)) \geq 0\}$, where $S = \{s_1, \ldots, s_n\}$, c_i^j are real-valued

[1] Intuitively, the Positivity problem asks for a given rational (or integer or real) matrix M, whether $(M^n)_{1,1} > 0$ for all n [54]. This problem (and its many variants) has been the subject of intense research over the last 10–15 years, see e.g. [55]. Yet, quite surprisingly, it still remains open in its full generality.

constants and N is the number of affine linear inequalities that define H. Our problem formally is given by the following query.

Problem Statement Given an MDP $\mathcal{M}$, initial distribution μ_0, and affine linear safe set H, (i) decide whether μ_0 is H-safe, and (ii) if yes, then synthesize a strategy for $\mathcal{M}$ which ensures safety.

Note that the problem strictly subsumes the special case when H is defined in terms of rational constants, and our approach aims to solve both problems. Also, note that Theorem 1 still applies, i.e. this "simplified" problem is Positivity-hard, too. We thus aim for a sound and *relatively complete* approach. Intuitively, this means that we restrict our search to a sub-space of possible solutions and within this space provide a complete answer. To give an intuition for the required reasoning, we provide an example safety query together with a manual proof.

Example 1. Consider our running example from Fig. 1. Suppose the initial distribution is $\mu_0 = \{A \mapsto \frac{1}{3}, B \mapsto \frac{1}{3}, C \mapsto \frac{1}{3}\}$ and (affine linear) $H = \{\mu \mid \mu(C) \geq \frac{1}{4}\}$. This safety query is satisfiable, by, e.g., choosing action b, as we show in the following. First, observe that the $i+1$-th distribution is $\mu_{i+1}(A) = \frac{1}{2} \cdot \mu_i(C)$, $\mu_{i+1}(B) = \mu_i(A)$, and $\mu_{i+1}(C) = \mu_i(B) + \frac{1}{2}\mu_i(C)$. Thus, we cannot directly prove by induction that $\mu_i(C) \geq \frac{1}{4}$, we also need some information about $\mu_i(B)$ or $\mu_i(A)$ to exclude, e.g., $\mu_i = \{A \mapsto \frac{3}{4}, C \mapsto \frac{1}{4}\}$, where μ_{i+1} would violate the safety constraint. We invite the interested reader to try to prove that μ_0 is indeed H-safe under the given strategy to appreciate the subtleties.

We proceed by proving that $\mu_i(C) \geq \frac{1}{4}$ and additionally $\mu_i(A) \leq \mu_i(C)$ by induction. The base case follows immediately, thus suppose that μ_i satisfies these constraints. For $\mu_{i+1}(A) \leq \mu_{i+1}(C)$ observe that $\mu_{i+1}(A) = \frac{1}{2}\mu_i(C)$ and $\mu_{i+1}(C) = \frac{1}{2}\mu_i(C) + \mu_i(B)$. Since $\mu_i(B) \geq 0$, the claim follows. To prove $\mu_{i+1}(C) \geq \frac{1}{4}$ observe that $\mu_i(A) \leq \frac{1}{2}$ since $\mu_i(A) \leq \mu_i(C)$ by induction hypothesis and distributions sum up to 1. Moreover, $\mu_{i+1}(C) = \mu_i(B) + \frac{1}{2}\mu_i(C) = \frac{1}{2}\mu_i(B) + \frac{1}{2} - \frac{1}{2}\mu_i(A)$ by again inserting the fact that distributions sum up to 1. Then, $\mu_{i+1}(C) = \frac{1}{2} - \frac{1}{2}\mu_i(A) + \frac{1}{2}\mu_i(B) \geq \frac{1}{2} - \frac{1}{2}\mu_i(A) \geq \frac{1}{2} - \frac{1}{4} \geq \frac{1}{4}$. △

Thus, already for rather simple examples the reasoning is non-trivial. To further complicate things, the structure of strategies can also be surprisingly complex:

Example 2. Again consider our running example from Fig. 1 with initial distribution $\mu_0 = \{A \mapsto \frac{3}{4}, B \mapsto \frac{1}{4}\}$ and safe set $H = \{\mu \mid \mu(B) = \frac{1}{4}\}$. This safety condition is indeed satisfiable, however the (unique) optimal strategy requires both infinite memory as well as randomization with arbitrarily small fractions! In step 1, we require choosing a with $\frac{2}{3}$ and b with $\frac{1}{3}$ to satisfy the safety constraint in the second step, getting $\mu_1 = \{A \mapsto \frac{1}{2}, B \mapsto \frac{1}{4}, C \mapsto \frac{1}{4}\}$. For step 2, we require choosing both a and b with probability $\frac{1}{2}$ each, yielding $\mu_2 = \{A \mapsto \frac{3}{8}, B \mapsto \frac{1}{4}, C \mapsto \frac{3}{8}\}$. Continuing this strategy, we obtain at step i that $\mu_i = \{A \mapsto \frac{1}{4} + \frac{1}{2^{i+1}}, B \mapsto \frac{1}{4}, C \mapsto \frac{1}{2} - \frac{1}{2^{i+1}}\}$ and action a is chosen with probability $1/(2^{i-1}+1)$, converging to 1. △

In the following, we provide two algorithms that handle both examples. Our first algorithm focusses on memoryless strategies, the second considers a certain type of infinite memory strategies. Essentially, the underlying idea is to automatically synthesize a strategy together with such inductive proofs of safety.

4 Proving Safety by Invariants

We now discuss our principled idea of proving safety by means of (inductive) invariants, taking inspiration from research on safety analysis in programs [20, 27]. We first show that considering strategies which are purely based on the current distribution over states are sufficient. Then, we show that inductive invariants are a *sound and complete* certificate for safety. Together, we obtain that an initial distribution is H-safe *if and only if* there exists an invariant set I and distribution strategy π such that (i) the initial distribution is contained in I, (ii) I is a subset of the safe set H, and (iii) I is inductive under π, i.e. if $\mu \in I$ then $\mathcal{M}^{\pi}(\mu) \in I$. In the following section, we then show how we search for invariants and distribution strategies *of a particular shape*.

4.1 Distribution Strategies

We show that *distribution strategies* $\pi : \Delta(S) \to \Pi_1$, yielding for each distribution over states a one-step strategy to take next, are sufficient for the problem at hand. More formally, we want to show that an H-safe distribution strategy exists if and only if there exists any H-safe strategy.

First, observe that distribution strategies are a special case of regular path strategies. In particular, for any given initial distribution, we obtain a uniquely determined stream of distributions as $\mu_{i+1} = \mathcal{M}^{\pi(\mu_i)}(\mu_i)$, i.e. the distribution μ_{i+1} is obtained by applying the one-step strategy $\pi(\mu_i)$ to μ_i. In turn, this lets us define the Markov strategy $\hat{\pi}_i(s) = \pi(\mu_i)(s)$. For simplicity, we identify distribution strategies with their induced path strategy.

Next, we argue that restricting to distribution strategies is sufficient.

Theorem 2. *An initial distribution μ_0 is H-safe if and only if there exists a distribution strategy π such that μ_0 is H-safe under π.*

Proof (Sketch). The full proof can be found in [4, Sec. 4.1]. Intuitively, only the "distribution" behaviour of a strategy is relevant and we can sufficiently replicate the behaviour of any safe strategy by a distribution strategy. □

In this way, each MDP corresponds to a (uncountably infinite) transition system $\mathcal{T}_{\mathcal{M}} = (\Delta(S), T)$ where $(\mu, \mu') \in T$ if there exists a one-step strategy π such that $\mu' = \mathcal{M}^{\pi}(\mu)$. Note that $\mathcal{T}_{\mathcal{M}}$ is a purely non-deterministic system, without any probabilistic behaviour. So, our decision problem is equivalent to asking whether the induced transition system $\mathcal{T}_{\mathcal{M}}$ can be controlled in a safe way. Note that $\mathcal{T}_{\mathcal{M}}$ is uncountably large and uncountably branching.

4.2 Distributional Invariants for MDP Safety

We now define distributional invariants in MDPs and show that they provide sound and complete certificates for proving initialized (and uninitialized) safety.

Distributional Invariants in MDPs. Intuitively, a distributional invariant is a set of probability distributions over MDP states that contains all probability distributions that can arise from applying a strategy to an initial probability distribution, i.e. the complete stream μ_i. Hence, similar to the safe set H, distributional invariants are also defined to be subsets of $\Delta(S)$.

Definition 1 (Distributional Invariants). *Let $\mu_0 \in \Delta(S)$ be a probability distribution over S and π be a strategy in $\mathcal{M}$. A set $I \subseteq \Delta(S)$ is said to be a* distributional invariant for μ_0 under π *if the sequence of probability distributions induced by applying the strategy π to the initial probability distribution μ_0 is contained in I, i.e. if $\mathcal{M}^\pi(\mu_0, i) \in I$ for each $i \geq 0$.*

A distributional invariant I is said to be inductive under π*, if we furthermore have that $\mathcal{M}^\pi(\mu) \in I$ holds for any $\mu \in I$, i.e. if I is "closed" under application of $\mathcal{M}^\pi$ to any probability distribution contained in I.*

Soundness and Completeness for MDP Safety. The following theorem shows that, in order to solve the initialized (and uninitialized) safety problem, one can equivalently search for a distributional invariant that is fully contained in H. Furthermore, it shows that one can without loss of generality restrict the search to inductive distributional invariants.

Theorem 3 (Sound and Complete Certificate). *Let $\mu_0 \in \Delta(S)$ be a probability distribution over S, π be a strategy in $\mathcal{M}$, and $H \subseteq \Delta(S)$ be a safe set. Then μ_0 is H-safe under π if and only if there exists an inductive distributional invariant I for μ_0 and π such that $I \subseteq H$.*

The proof can be found in [4, Sec. 4.2].

Thus, in order to solve the initialized safety problem for μ_0, it suffices to search for (i) a strategy π and (ii) an inductive distributional invariant I for μ_0 and π such that $I \subseteq H$. On the other hand, in order to solve the uninitialized safety problem, it suffices to search for (i) an initial probability distribution μ_0, (ii) strategy π, and (iii) an inductive distributional invariant I for μ_0 and π such that $I \subseteq H$. In the following, we provide a fully automated, sound and *relatively* complete method of deciding the existence of such an invariant and strategy.

5 Algorithms for Distributional Invariant Synthesis

We now present two algorithms for automated synthesis of strategies and inductive distributional invariants towards solving distribution safety problems in MDPs. The two algorithms differ in the kind of strategies they consider and, as a consequence of differences in the involved expressions, also in their completeness guarantees. For readability, we describe the algorithms in their basic

form applied to the initialized variant of the safety problem and discuss further extensions in Sect. 6. In particular, our approach is also directly applicable to the uninitialized variant, as we describe there.

We say that an inductive distributional invariant is *affine* if it can be specified in terms of (non-strict) affine inequalities, which we formalize below. Both algorithms jointly synthesize a strategy and an affine inductive distributional invariant by employing a *template-based synthesis* approach. In particular, they fix symbolic templates for each object that needs to be synthesized, encode the defining properties of each object as constraints over unknown template variables, and solve the system of constraints by reduction to the existential first-order theory of the reals.

For example, a template for an affine linear constraint on distributions $\Delta(S)$ is given by $\mathsf{aff}(\mu) = (c_0 + c_1 \cdot \mu(s_1) + \cdots + c_n \cdot \mu(s_n) \geq 0)$. Here, the variables c_0 to c_n, written in grey for emphasis, are the *template variables*. For fixed values of these variables the expression aff is a concrete affine linear predicate over distributions. Thus, we can ask questions like "Do there exist values for c_i such that for all distributions μ we have that $\mathsf{aff}(\mu)$ implies $\mathsf{aff}(\mathcal{M}^\pi(\mu))$?". This is a sentence in the theory of reals – however with quantifier alternation. As a next step, template-based synthesis approaches then employ various quantifier elimination techniques to convert such expressions into equisatisfiable sentences in, e.g., the existential theory of reals, which is decidable in PSPACE [15].

Difference between the Algorithms. Our two algorithms differ in their applicability and the kind of completeness guarantees that they provide. In terms of applicability, the first algorithm only considers *memoryless* strategies, while the second algorithm searches for *distribution* strategies specified as fractions of affine linear expressions. (We discuss an extension to rational functions in Sect. 6.) In terms of completeness guarantees, the first algorithm is *(relatively) complete* in the sense that it is guaranteed to compute a memoryless strategy and an affine inductive distributional invariant that prove safety *whenever they exist*. In contrast, the second algorithm does not provide the same level of completeness.

Notation. In what follows, we write $\equiv$ to denote (syntactic) equivalence of expressions, to distinguish from relational symbols used inside these expressions, such as "=". For example $\Phi(x) \equiv x = 0$ means that $\Phi(x)$ is the predicate $x = 0$. Moreover, $(x_1, \ldots, x_n)$ denotes a symbolic probability distribution over the state space $S = (s_1, \ldots, s_n)$, where x_i is a symbolic variable that encodes the probability of the system being in s_i. We use boldface notation $\boldsymbol{x} = (x_1, \ldots, x_n)$ to denote the vector of symbolic variables. Thus, the above example would be written $\mathsf{aff}(\boldsymbol{x}) \equiv c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n \geq 0$. Since we often require vectors to represent a distribution, we write $\boldsymbol{x} \in \Delta(S)$ as abbreviation for the predicate $\bigwedge_{i=1}^{n}(0 \leq x_i \leq 1) \wedge (\sum_{i=1}^{n} x_i = 1)$.

Algorithm Input and Assumptions. Both algorithms take as input an MDP $\mathcal{M} = (S, Act, \delta)$ with $S = \{s_1, \ldots, s_n\}$. They also take as input a safe set $H \subseteq \Delta(S)$.

We assume that H is specified by a boolean predicate over n variables as a logical conjunction of $N_H \in \mathbb{N}_0$ *affine* inequalities, and that it has the form

$$H(\boldsymbol{x}) \equiv (\boldsymbol{x} \in \Delta(S)) \wedge \bigwedge\nolimits_{i=1}^{N_H} (h^i(\boldsymbol{x}) \geq 0),$$

where the first term imposes that $\boldsymbol{x}$ is a probability distribution over S and $h^i(\boldsymbol{x}) = h^i_0 + h^i_1 \cdot x_1 + \cdots + h^i_n \cdot x_n$ is an affine expression over $\boldsymbol{x}$ with real-valued coefficients h^i_j for each $i \in [N_H]$ and $j \in \{0, \ldots, n\}$. (Note that h^i_j are not template variables but fixed values, given as input.) Next, the algorithms take as input an initial probability distribution $\mu_0 \in \Delta(S)$. Finally, the algorithms also take as input technical parameters. Intuitively, these describe the size of used *symbolic templates*, explained later. For the remainder of the section, fix an initialized safety problem, i.e. an $\mathcal{M}$, safe set H of the required form, and an initial distribution μ_0.

5.1 Synthesis of Affine Invariants and Memoryless Strategies

We start by presenting our first algorithm, which synthesizes memoryless strategies and affine inductive distributional invariants. We refer to this algorithm as AlgMemLess. The algorithm proceeds in the following four steps:

1. *Setting up Templates.* The algorithm fixes symbolic templates for the memoryless strategy π and the affine inductive distributional invariant I. Note that the values of the symbolic template variables at this step are *unknown* and are to be computed in subsequent steps.
2. *Constraint Collection.* The algorithm collects the constraints which encode that π is a (memoryless) strategy, that I contains the initial probability distribution μ_0, that I is an inductive distributional invariant with respect to π and μ_0, and that I is contained within H. This step yields a system of affine constraints over symbolic template variables that contain universal and existential quantifiers.
3. *Quantifier Elimination.* The algorithm eliminates universal quantifiers from the above constraints to reduce it to a system of purely existentially quantified system of polynomial constraints over the symbolic template variables. Concretely, the first algorithm achieves this by application of *Farkas' lemma.*
4. *Constraint Solving.* The algorithm solves the resulting system of constraints by using an off-the-shelf solver to compute concrete values for symbolic template variables specifying the strategy π and invariant I.

We now describe each step in detail.

Step 1: Setting up Templates. The algorithm sets templates for π and I as follows:

– Since this algorithm searches for memoryless strategies, the probability of taking an action a_j in state s_i is always the same, independent of the current distribution. Hence, our template for π consists of a symbolic template variable p_{s_i,a_j} for each $s_i \in S$, $a_j \in Act(s_i)$. We write $p_{s_i,\circ} = (p_{s_i,a_1}, \ldots, p_{s_i,a_m})$ to refer to the corresponding distribution in state s_i.

– The template of I is given by a boolean predicate specified by a conjunction of N_I affine inequalities, where N_I is the *template size* and is an algorithm parameter. In particular, the template of I looks as follows:

$$I(\boldsymbol{x}) \equiv (\boldsymbol{x} \in \Delta(S)) \wedge \bigwedge\nolimits_{i=1}^{N_I} (a_0^i + a_1^i \cdot x_1 + \cdots + a_n^i \cdot x_n \geq 0).$$

The first predicate enforces that I only contains vectors that define probability distributions over S.

Step 2: Constraint Collection. We now collect the constraints over symbolic template variables which encode that π is a memoryless strategy, that I contains the initial distribution μ_0, that I is an inductive distributional invariant under π, and that I is contained in H.

– For π to be a strategy, we only need to ensure that each $p_{s_i,\circ}$ is a probability distribution over the set of available actions at every state s_i. Thus, we set

$$\Phi_{\text{strat}} \equiv \bigwedge\nolimits_{i=1}^{n} \left(p_{s_i,\circ} \in \Delta(Act(s_i))\right).$$

– For I to be a distributional invariant for π and μ_0 as well as to be inductive, it suffices to enforce that I contains μ_0 and that I is closed under application of π. Thus, we collect two constraints:

$$\Phi_{\text{initial}} \equiv I(\mu_0) \equiv \bigwedge\nolimits_{i=1}^{N_I} (a_0^i + a_1^i \cdot \mu_0^1 + \ldots a_n^i \cdot \mu_0^n \geq 0), \text{ and}$$
$$\Phi_{\text{inductive}} \equiv (\forall \boldsymbol{x} \in \mathbb{R}^n.\ I(\boldsymbol{x}) \implies I(\text{step}(\boldsymbol{x}))),$$

where $\text{step}(\boldsymbol{x})(x_i) = \sum_{s_k \in S, a_j \in Act(s_k)} p_{s_k,a_j} \cdot \delta(s_k, a_j, s_i) \cdot x_j$ yields the distribution after applying one step of the strategy induced by Φ_{strat} to $\boldsymbol{x}$.

– For I to be contained in H, we enforce the constraint:

$$\Phi_{\text{safe}} \equiv (\forall \boldsymbol{x} \in \mathbb{R}^n.\ I(\boldsymbol{x}) \implies H(\boldsymbol{x})).$$

Step 3: Quantifier Elimination. Constraints Φ_{strat} and Φ_{initial} are purely existentially quantified over symbolic template variables, thus we can solve them directly. However, $\Phi_{\text{inductive}}$ and Φ_{safe} contain both universal and existential quantifiers, which are difficult to handle. In what follows, we show how the algorithm translates these constraints into equisatisfiable *purely existentially quantified* constraints. In particular, our translation exploits the fact that both $\Phi_{\text{inductive}}$ and Φ_{safe} can, upon splitting the conjunctions on the right-hand side of implications into conjunctions of implications, be expressed as conjunctions of constraints of the form

$$\forall \boldsymbol{x} \in \mathbb{R}^n.\ (\text{affexp}_1(\boldsymbol{x}) \geq 0) \wedge \cdots \wedge (\text{affexp}_N(\boldsymbol{x}) \geq 0) \implies (\text{affexp}(\boldsymbol{x}) \geq 0).$$

Here, each $\text{affexp}_i(\boldsymbol{x})$ and $\text{affexp}(\boldsymbol{x})$ is an affine expression over $\boldsymbol{x}$ whose affine coefficients are either concrete real values or symbolic template variables.

In particular, we use Farkas' lemma [31] to remove universal quantification and translate the constraint into an equisatisfiable existentially quantified system of constraints over the symbolic template variables, as well as fresh auxiliary variables that are introduced by the translation. For completeness, we briefly recall (a strengthened and adapted version of) Farkas' lemma.

Lemma 1 ([31,37]). *Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite set of real-valued variables, and consider the following system of $N \in \mathbb{N}$ affine inequalities over $\mathcal{X}$:*

$$\Phi : \begin{cases} c_0^1 + c_1^1 \cdot x_1 + \cdots + c_n^1 \cdot x_n \geq 0 \\ \vdots \\ c_0^N + c_1^N \cdot x_1 + \cdots + c_n^N \cdot x_n \geq 0 \end{cases}.$$

Suppose that Φ is satisfiable. Then Φ entails an affine inequality $\phi \equiv c_0 + c_1 \cdot x_1 + \cdots + c_n \cdot x_n$, i.e. $\Phi \implies \phi$, if and only if ϕ can be written as a non-negative *linear combination of affine inequalities in Φ, i.e. if and only if there exist $y_1, \ldots, y_n \geq 0$ such that $c_1 = \sum_{j=1}^{N} y_j \cdot c_1^j$, $\ldots$, $c_n = \sum_{j=1}^{N} y_j \cdot c_n^j$.*

Note that, for any implication appearing in $\Phi_{\text{inductive}}$ and Φ_{safe}, the system of constraints on the left-hand side is simply $I(\boldsymbol{x})$, and the satisfiability of $I(\boldsymbol{x})$ is enforced by Φ_{initial}. Hence, we may apply Farkas lemma to translate each constraint with universal quantification into an equivalent purely existentially quantified constraint. In particular, for any constraint of the form

$$\forall \boldsymbol{x} \in \mathbb{R}^n. \, (\text{affexp}_1(\boldsymbol{x}) \geq 0) \wedge \cdots \wedge (\text{affexp}_N(\boldsymbol{x}) \geq 0) \implies (\text{affexp}(\boldsymbol{x}) \geq 0),$$

we introduce fresh template variables $y_1, \ldots, y_N$ and translate it into the system of purely existentially quantified constraints

$$(y_1 \geq 0) \wedge \cdots \wedge (y_N \geq 0) \wedge (\text{affexp}(\boldsymbol{x}) \equiv_F y_1 \cdot \text{affexp}_1(\boldsymbol{x}) + \cdots + y_N \cdot \text{affexp}_N(\boldsymbol{x})).$$

Here, we use $\text{affexp}(\boldsymbol{x}) \equiv_F y_1 \cdot \text{affexp}_1(\boldsymbol{x}) + \cdots + y_N \cdot \text{affexp}_N(\boldsymbol{x})$ to denote the set of $n+1$ equalities over the symbolic template variable and $y_1, \ldots, y_N$ which equate the constant coefficients as well as the linear coefficients of each x_i on two sides of the equivalence, i.e. exactly those equalities which we obtain from applying Farkas' lemma. We highlight that the expressions affexp are only affine linear for *fixed* existentially quantified variables, i.e. they are in general quadratic.

Step 4: Constraint Solving. Finally, we feed the resulting system of existentially quantified polynomial constraints over the symbolic template variables as well as the auxiliary variables introduced by applying Farkas' lemma to an off-the-shelf constraint solver. If the solver outputs a solution, we conclude that the computed invariant I is an inductive distributional invariant for the strategy π and initial distribution μ_0, and that I is contained in H. Therefore, by Theorem 3, we conclude that μ_0 is H-safe under π.

$$\Phi_{\text{init}} : c_0 + c_1 \cdot \tfrac{1}{3} + c_2 \cdot \tfrac{1}{3} + c_3 \cdot \tfrac{1}{3} \geq 0$$
$$\Phi_{\text{safe}} : (c_0 + c_1 \cdot A + c_2 \cdot B + c_3 \cdot C \geq 0) \implies C \geq \tfrac{1}{4}$$
$$\Phi_{\text{inductive}} : \begin{array}{c}(c_0 + c_1 \cdot A + c_2 \cdot B + c_3 \cdot C \geq 0) \implies \\ c_0 + c_1 \cdot (A \cdot p_{A,a_1} + \tfrac{1}{2}C) + c_2 \cdot A \cdot p_{A,a_2} + c_3 \cdot (B + \tfrac{1}{2}C) \geq 0\end{array}$$
$$\Phi_{\text{strat}} : p_{A,a_1} \geq 0 \quad p_{A,a_2} \geq 0 \quad p_{A,a_1} + p_{A,a_2} = 1$$

Fig. 2. List of constraints generated in Step 2 for Example 1 with $N_I = 1$. The uppercase letters correspond to variables indicating the distribution in these states, i.e. A refers to $\mu(A)$. These also are the universally quantified variables, which will be handled by the quantifier elimination in Step 3. The template variables are written in grey. For readability, we omit the constraints required for state distributions $\mu \in \Delta(S)$, i.e. $A \geq 0$ etc. The actual query sent to the solver in Step 4 after quantifier elimination comprises 27 constraints with 21 variables.

Theorem 4. Soundness*: Suppose* AlgMemLess *returns a memoryless strategy* π *and an affine inductive distributional invariant* I*. Then,* μ_0 *is* H*-safe under* π*.*

Completeness*: If there exist a memoryless strategy* π *and an affine inductive distributional invariant* I *such that* $I \subseteq H$ *and* μ_0 *is* H*-safe under* π*, then there exists a minimal value of the template size* $N_I \in \mathbb{N}$ *such that* π *and* I *are produced by* AlgMemLess*.*

Complexity*: The runtime of* AlgMemLess *is in PSPACE in the size of the MDP, the encoding of the safe set* H *and the template size parameter* $N_I \in \mathbb{N}$*.*

The proof can be found in [4, Sec. 5.1]. We comment on the PSPACE upper bound on the complexity of AlgMemLess. The upper bound holds since the application of Farkas' lemma reduces synthesis to solving a sentence in the existential first-order theory of the reals and since the size of the sentence is polynomial in the sizes of the MDP, the encoding of the safe set H and the invariant template size N_i. However, it is unclear whether the resulting constraints could be solved more efficiently, and the best known upper bound on the time complexity of algorithms for template-based affine inductive invariant synthesis in programs is also PSPACE [8,27]. Designing more efficient algorithms for solving constraints of this form would lead to better algorithms both for the safety problem studied in this work and for template-based affine inductive invariant synthesis in programs.

Example 3. For completeness, we provide the constraints generated in Step 2 for Example 1 with $N_I = 1$ for readability, i.e. our running example Fig. 1 with $\mu_0 = \{A \mapsto \frac{1}{3}, B \mapsto \frac{1}{3}, C \mapsto \frac{1}{3}\}$ and $H = \{\mu \mid \mu(C) \geq \frac{1}{4}\}$, in Fig. 2.

To conclude this section, we emphasize that our algorithm *simultaneously* synthesizes both the invariant and the witnessing strategy, which is the key component to achieve relative completeness.

5.2 Synthesis of Affine Invariants and General Strategies

We now present our second algorithm, which additionally synthesizes *distribution strategies* (of a particular shape) together with an affine inductive distributional invariant. We refer to it as AlgDist. The second algorithm proceeds in the analogous four steps as the first algorithm, AlgMemLess. Hence, in the interest of space, we only discuss the differences compared to AlgMemLess.

Step 1: Setting up Templates. The algorithm sets up templates for π and I. The template for I is defined analogously as in Sect. 5.1. However, as we now want to search for a strategy π that need not be memoryless but instead may depend on the current distribution, we need to consider a more general template. In particular, the template for the probability p_{s_i,a_j} of taking an action a_j in state s_i is no longer a constant value. Instead, $p_{s_i,a_j}(\boldsymbol{x})$ is a function of the probability distribution $\boldsymbol{x}$ of the current state of the MDP, and we define its template to be a quotient of two affine expressions for each $s_i \in S$ and $a_j \in Act(s_i)$:

$$p_{s_i,a_j}(\boldsymbol{x}) \equiv \frac{\mathrm{num}(s_i,a_j)(\boldsymbol{x})}{\mathrm{den}(s_i)(\boldsymbol{x})} \equiv \frac{r_0^{i,j} + r_1^{i,j} \cdot x_1 + \cdots + r_n^{i,j} \cdot x_n}{s_0^i + s_1^i \cdot x_1 + \cdots + s_n^i \cdot x_n}.$$

(In Sect. 6, we discuss how to extend our approach to polynomial expressions for numerator and denominator, i.e. rational functions.) Note that the coefficients in the numerator depend both on the state s_i and the action a_j, whereas the coefficients in the denominator depend only on the state s_i. This is because we only use the affine expression in the denominator as a normalization factor to ensure that p_{s_i,a_i} indeed defines a probability.

Step 2: Constraint Collection. As before, the algorithm now collects the constraints over symbolic template variables which encode that π is a strategy, that I is an inductive distributional invariant, and that I is contained in H. The constraints Φ_{initial}, $\Phi_{\text{inductive}}$, and Φ_{safe} are defined analogously as in Sect. 5.1, with the necessary adaptation to $\mathrm{step}(\boldsymbol{x})$. For the strategy constraint Φ_{strat} we now need to take additional care to ensure that each quotient template defined above does not induce division by 0 and that these values indeed correspond to a distribution over the available actions. We ensure this by the following constraint:

$$\Phi_{\text{strat}} \equiv \forall \boldsymbol{x} \in \mathbb{R}^n.\, I(\boldsymbol{x}) \implies \bigwedge\nolimits_{i=1}^{n} \left(\begin{array}{l} \bigwedge\nolimits_{a_j \in Act(s_i)} \mathrm{num}(s_i,a_j)(\boldsymbol{x}) \geq 0 \wedge \\ \mathrm{den}(s_i)(\boldsymbol{x}) \geq 1 \wedge \\ \sum\nolimits_{a_j \in Act(s_i)} \mathrm{num}(s_i,a_j)(\boldsymbol{x}) = \mathrm{den}(s_i)(\boldsymbol{x}). \end{array} \right).$$

The first two constraints ensure that all quantities are positive and we never divide by 0. The third means that the numerators sum up to the denominator. Together, this ensures the desired result, i.e. $p_{s_i,\circ}(\boldsymbol{x}) \in \Delta(Act(s_i))$ whenever $\boldsymbol{x} \in \Delta(S)$. Note that the ≥ 1 constraint for the denominator can be replaced by an arbitrary constant > 0, since we can always rescale all involved coefficients.

Step 3: Quantifier Elimination. The constraints Φ_{strat}, Φ_{initial}, and Φ_{safe} can be handled analogously to Sect. 5.1. In particular, by applying Farkas' lemma these can be translated into an equisatisfiable purely existentially quantified system of polynomial constraints, and our algorithm applies this translation.

However, the constraint $\Phi_{\text{inductive}}$ now involves quotients of affine expressions: Upon splitting the conjunction on the right-hand side of the implication in $\Phi_{\text{inductive}}$ into a conjunction of implications, the inequalities on the right-hand side of these implications contain templates for strategy probabilities $p_{s_i,a_j}(\boldsymbol{x})$. The algorithm removes the quotients by multiplying both sides of the inequality by denominators of each quotient. (Recall that each denominator is positive by the constraint Φ_{strat}.) This results in the multiplication of symbolic affine expressions, hence $\Phi_{\text{inductive}}$ becomes a conjunction of implications of the form

$$\forall \boldsymbol{x} \in \mathbb{R}^n. (\text{affexp}_1(\boldsymbol{x}) \geq 0) \wedge \cdots \wedge (\text{affexp}_N(\boldsymbol{x}) \geq 0) \implies (\text{polyexp}(\boldsymbol{x}) \geq 0).$$

Here, each $\text{affexp}_i(\boldsymbol{x})$ is an affine expression over $\boldsymbol{x}$, but $\text{polyexp}(\boldsymbol{x})$ is now a polynomial expression over $\boldsymbol{x}$. Hence we cannot apply a Farkas' lemma-style result to remove universal quantifiers.

Instead, we motivate our translation by recalling Handelman's theorem [38], which characterizes *strictly* positive polynomials over a set of affine inequalities. It will allow us to soundly translate $\Phi_{\text{inductive}}$ into an existentially quantified system of constraints over the symbolic template variables, as well as fresh auxiliary variables that are introduced by the translation.

Theorem 5 ([38]). *Let $\mathcal{X} = \{x_1, \ldots, x_n\}$ be a finite set of real-valued variables, and consider the following system of $N \in \mathbb{N}$ non-strict affine inequalities over $\mathcal{X}$:*

$$\Phi : \begin{cases} c_0^1 + c_1^1 \cdot x_1 + \cdots + c_n^1 \cdot x_n \geq 0 \\ \vdots \\ c_0^N + c_1^N \cdot x_1 + \cdots + c_n^N \cdot x_n \geq 0 \end{cases}.$$

Let $\text{Prod}(\Phi) = \{\prod_{i=1}^{t} \phi_i \mid t \in \mathbb{N}_0, \phi_i \in \Phi\}$ be the set of all products of finitely many affine expressions in Φ, where the product of 0 affine expressions is a constant expression 1. Suppose that Φ is satisfiable and that $\{\boldsymbol{y} \mid \boldsymbol{y} \models \Phi\}$, the set of values satisfying Φ, is topologically compact, i.e. closed and bounded. Then Φ entails a polynomial inequality $\phi(\boldsymbol{x}) > 0$ if and only if ϕ can be written as a non-negative linear combination of finitely many products in $\text{Prod}(\Phi)$, i.e. if and only if there exist $y_1, \ldots, y_n \geq 0$ and $\phi_1, \ldots, \phi_n \in \text{Prod}(\Phi)$ such that $\phi = y_1 \cdot \phi_1 + \cdots + y_n \cdot \phi_n$.

Notice that we cannot directly apply Handelman's theorem to a constraint

$$\forall \boldsymbol{x} \in \mathbb{R}^n. (\text{affexp}_1(\boldsymbol{x}) \geq 0) \wedge \cdots \wedge (\text{affexp}_N(\boldsymbol{x}) \geq 0) \implies (\text{polyexp}(\boldsymbol{x}) \geq 0),$$

since the polynomial inequality on the right-hand-side of the implication is non-strict whereas the polynomial inequality in Handelman's theorem is strict. However, the direction needed for the soundness of translation holds even with the

non-strict polynomial inequality on the right-hand side. In particular, it clearly holds that if polyexp can be written as a non-negative linear combination of finitely many products of affine inequalities, then polyexp is non-negative whenever all affine inequalities are non-negative. Hence, we may use the translation in Handelman's theorem to translate each implication in $\Phi_{\text{inductive}}$ into a system of purely existentially quantified constraints.

As Handelman's theorem does not impose a bound on the number of products of affine expressions that might appear in the translation, we *parametrize* the algorithm with an upper bound K on the maximal number of affine inequalities appearing in each product. To that end, we define $\text{Prod}_K(\Phi) = \{\prod_{i=1}^{t} \phi_i \mid 0 \leq t \leq K, \phi_i \in \Phi\}$. Let $M_K = |\text{Prod}_K(\Phi)|$ be the total number of such products and $\text{Prod}_K(\Phi) = \{\phi_1, \ldots, \phi_{M_K}\}$. Then, for any constraint of the form

$$\forall \boldsymbol{x} \in \mathbb{R}^n. (\text{affexp}_1(\boldsymbol{x}) \geq 0) \wedge \cdots \wedge (\text{affexp}_N(\boldsymbol{x}) \geq 0) \implies (\text{polyexp}(\boldsymbol{x}) \geq 0),$$

we introduce fresh template variables $y_1, \ldots, y_{M_K}$ and translate it into the system of purely existentially quantified constraints

$$(y_1 \geq 0) \wedge \cdots \wedge (y_N \geq 0) \wedge (\text{polyexp}(\boldsymbol{x}) \equiv_H y_1 \cdot \phi_1(\boldsymbol{x}) + \cdots + y_{M_K} \cdot \phi_{M_K}(\boldsymbol{x})).$$

Here, $\text{polyexp}(\boldsymbol{x}) \equiv_H y_1 \cdot \phi_1(\boldsymbol{x}) + \cdots + y_{M_K} \cdot \phi_{M_K}(\boldsymbol{x})$ denotes the set of equalities over template variables and $y_1, \ldots, y_{M_K}$ which equate the constant coefficients as well as the coefficients of each monomial over $\{x_1, \ldots, x_k\}$ of degree at most K on two sides of the equivalence, as specified by Handelman's theorem.

While our translation into a purely existentially quantified constraints is not complete due to the non-strict polynomial inequality and due to the parametrization by K, Handelman's theorem justifies the translation as it indicates that the translation is "close to complete" for sufficiently large values of K.

Step 4: Constraint Solving. This step is analogous to Sect. 5.1 and we use an off-the-shelf polynomial constraint solver to handle the resulting system of purely existentially quantified polynomial constraints. If the solver outputs a solution, we conclude that the computed I is an inductive distributional invariant for the computed strategy π and initial distribution μ_0, and that I is contained in H. Therefore, by Theorem 3, we conclude that μ_0 is H-safe under π.

Theorem 6. Soundness*: Suppose* AlgDist *returns a strategy* π *and an affine inductive distributional invariant* I*. Then,* π *is* H*-safe for* μ_0*.*

Complexity*: For any fixed parameter* $K \in \mathbb{N}$*, the runtime of* AlgDist *is in PSPACE in the size of the MDP and the template size parameter* $N_I \in \mathbb{N}$*.*

The proof can be found in [4, Sec. 5.2].

6 Discussion, Extensions, and Variants

With our two algorithms in place, we remark on several interesting details and possibilities for extensions.

Polynomial Expressions. Our second algorithm can also be extended to synthesizing *polynomial* inductive distributional invariants, i.e. instead of defining the invariant I through a conjunction of affine linear expressions we could synthesize polynomial expressions such as $x_1^2 + x_2 \cdot x_3 \leq 0.5$. This can be achieved by using Putinar's Positivstellensatz [59] instead of Handelman's theorem in Step 3. This technique has recently been used for generating polynomial inductive invariants in programs in [20], and our translation in Step 3 can be analogously adapted to synthesize polynomial inductive distributional invariants up to a specified degree. In the same way, instead of requiring that H is given as a conjunction of affine linear constraints, we can also handle the case of polynomial constraints. The same holds true for the probabilities of choosing certain actions $p_{s_i,a_j}(\boldsymbol{x})$. While we have defined these as fractions of affine linear expressions, we could replace them with rational functions, which we chose to exclude for sake of readability.

Uninitialized and Restricted Initial Case. We remark that we can directly incorporate the uninitialized case in our algorithm. In particular, instead of requiring that $I(\mu_0)$ holds for the concretely given initial values, we can instead existentially quantify over the values of $\mu_0(s_i)$ and add the constraint that μ_0 is a distribution, i.e. $\mu_0(s_i) \in \Delta(S)$. This does not add universal quantification, thus we do not need to apply any quantifier elimination for these variables. This also subsumes and generalizes the ideas of [5], which observes that checking whether a fixpoint of the transition dynamics lies within H is sufficient. Choosing $I = \{\mu^*\}$ where μ^* is such a fixpoint satisfies all of our constraints. See [4, Sec. 6] for details.

Our algorithm is also able to handle the "intermediate" case, as follows. The uninitialized case leaves absolute freedom in the choice of initial distribution, while the initialized case concretely specifies one initial distribution. Here, we could as well impose *some* constraints on the initial distribution without fixing it completely, i.e. ask whether there exists an H-safe initial distribution μ_0 which satisfies a predicate Φ_{init}. If Φ_{init} is a conjunction of affine linear constraints, we can directly handle this query, too. Note that both initialized and uninitialized are special cases thereof.

Non-Inductive Initial Steps. Instead of requiring to synthesize an invariant which contains the initial distribution, we can explicitly write down the first k distributions and only then require an invariant and strategy to be found. More concretely, the set of distributions that can be achieved in a given step k while remaining in H can be explicitly computed, denote this set as Δ^k. For a different perspective, this describes the set of states reachable in $\mathcal{T}_{\mathcal{M}}$ within k steps and corresponds to "unrolling" the MDP for a fixed number of steps. This then goes hand in hand with the above "restricted initial case", where we ask whether there exists an H-safe distribution in Δ^k. We conjecture that this could simplify the search for distributional invariants for systems which have a lot of "transient" behaviour, as observed in searching for invariants for state reachability [11].

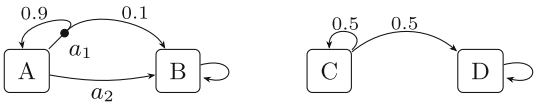

Fig. 3. Our Split toy example. The MDP comprises two disconnected parts. Probability mass flows from A to B and from C to D under all strategies.

7 Implementation and Evaluation

While the main focus of our contribution lies on the theory, we validate the applicability through an unoptimized prototype implementation. We implemented our approach in Python 3.10, using `SymPy` 1.11 [50] to handle and simplify symbolic expressions, and `PySMT` 0.9 [36] to abstract communication with constraint solvers. We use `z3` 4.8 [53] and `mathsat` 5.6 [26] as back-ends. Our experiments were executed on consumer hardware (AMD Ryzen 3600 CPU with 16 GB RAM).

Caveats. While the existential (non-linear) theory of the reals is known to be decidable, practical algorithms are less explored than, for example, SAT solving. In particular, runtimes are quite sensitive to minor changes in the input structure and initial randomization (many solvers apply randomized algorithms). We observed differences of several orders of magnitude (going from seconds to hours) simply due to restarting the computation (leading to different initial seeds). Similarly, by strengthening the antecedents of implications by known facts, we also observed significant improvements. Concretely, given that we have constraints of the form $I(\boldsymbol{x}) \implies H(\boldsymbol{x})$ and $I(\boldsymbol{x}) \implies \Phi(\boldsymbol{x})$, we observed that changing the second constraint to $I(\boldsymbol{x}) \wedge H(\boldsymbol{x}) \implies \Phi(\boldsymbol{x})$ would drastically improve the runtime even though the two are semantically equivalent.

This suggests that both improvements of our implementation as well as further work on constraint solvers are likely to have a significant impact on the runtime.

Models. Aside from our running example of Fig. 1, which we refer to as Running here, we consider two further toy examples.

The first model, called Chain, is a Markov chain defined as follows: We consider the states $S = \{s_1, \ldots, s_{10}\}$ and set $\delta(s_i) = \{s_{i+1} \mapsto 1\}$ for all $i < 10$ and $\delta(s_{10}) = \{s_9 \mapsto \frac{1}{2}, s_{10} \mapsto \frac{1}{2}\}$. The initial distribution is given as $\mu_0(s_i) = \frac{1}{10}$ for all $s_i \in SS$ and the safe set by $H = \{\mu(s_{10}) \geq \frac{1}{10}\}$. We are mainly interested in this model to investigate demonstrate applicability to "larger" systems.

The second model, called Split, is an MDP which actually comprises two independent subsystems. We depict the model in Fig. 3. The initial distribution is $\mu_0 = \{A \mapsto \frac{1}{2}, C \mapsto \frac{1}{2}\}$ and the safe set $H = \{\mu(A) + \mu(D) \geq \frac{1}{2}\}$. This aims to explore both disconnected models as well as a safe set which imposes a constraint on multiple states at once. In particular, observe that initially $\mu_0(D) = 0$ but $\mu_i(D)$ converges to 1 while $\mu_i(A)$ converges to 0, even if choosing action a_1. Thus, the invariant needs to identify the simultaneous flow from A to B and C to D.

Table 1. Overview of our results for the five considered models. From left to right, we list the name of the model, the runtime, and size of the invariant, followed by the number of variables, constraints, and total size of the query passed to the constraint solvers. For Running, we provided additional hints to the solver to achieve a more consistent runtime, indicated by the dagger symbol.

Model	Runtime	N_I	#Var.	#Constr.	Size.
Running	3s†	3	92	123	849
Chain	10s	2	69	82	666
Split	3s	3	60	69	571
PageRank	3s	2	44	52	536
Insulin-^{131}I	2s	2	44	52	476

Table 2. The invariants and strategies computed for our models. We omit the invariants for the two real-world scenarios since they are too large to fit.

Model	Computed Invariant and Strategy
Running	$\{A \geq \frac{1}{4}, B = \frac{1}{4}\}$ $\quad \pi(\mu) = \{a_1 \mapsto \frac{1}{4 \cdot \mu(A)}, a_2 \mapsto \frac{4 \cdot \mu(A) - 1}{4 \cdot \mu(A)}\}$
Chain	$\{s_9 + s_{10} \geq \frac{1}{5}, s_{10} \geq \frac{1}{10}\}$ $\quad \pi = \emptyset$ (Markov chain)
Split	$\{B \leq D, A + B \geq C + D, 3 \cdot (C + D) - (A + B) \geq 1\}$ $\quad \pi = \{a \mapsto 1\}$

We additionally consider two examples from the literature, namely the PageRank example from [1, Fig. 3], based on [51], and Insulin-^{131}I, a pharmacokinetics system [1, Example 2], based on [17]. Both are Markov chains.

Results. We summarize our findings briefly in Table 1. We again underline that not too much attention should be put on runtimes, since they are very sensitive to minimal changes in the model. The evaluation is mainly intended to demonstrate that our methods are actually able to provide results. For completeness, we report the size of the invariant N_I and the size of the constraint problem in terms of number of variables, constraints, and operations inside these constraints. We also provide the invariants and strategy identified by our method in Table 2. Note that for Running we used AlgDist, while the other two examples are handled by AlgMemLess. For Running, we observed a significant dependence on the initialization of the solvers. Thus we added several "hints", i.e. known correct values for some variables. (To be precise, we set the value for eight of the 92 variables.)

Discussion. We remark two related points: Firstly, we observe that very often most of the involved auxiliary variables introduced by the quantifier elimination have a value of zero. Thus, a potential optimization is to explicitly set most such variables to zero, check whether the formula is satisfiable, and, if not, gradually remove these constraints either at random or guided by unsat-cores if available (i.e. clauses which are the "reason" for unsatisfiability). Moreover, we observed

significant differences between the solvers: While `z3` seems to be much quicker to identify unsatisfiability, `mathsat` usually is better at finding satisfying assignments. Hence, using both solvers in tandem seems to be very beneficial.

8 Conclusion

We developed a framework for defining certificates for safety objectives in MDPs as distributional inductive invariants. Using this, we came up with two algorithms that synthesize linear/affine invariants and corresponding memoryless/general strategies for safety in MDPs. To the best of our knowledge this is the first time the template-based invariant approach, already known to be successful for programs, has been applied to synthesis strategies in MDPs for distributional safety properties. Our experimental results show that affine invariants are sufficient for many interesting examples. However, the second approach can be lifted to synthesize polynomial invariants, and hence potentially, a large set of MDPs. Exploring this could be a future line of work. It would also be interesting to explore how one can automate distributional invariant synthesis if the safe set H is specified in terms of both strict and non-strict inequalities. Finally, in terms of applicability, we would like to apply this approach to solve more benchmarks and problems, e.g., to synthesize risk-aware strategies for MDPs [46,49].

References

1. Agrawal, M., Akshay, S., Genest, B., Thiagarajan, P.S.: Approximate verification of the symbolic dynamics of Markov chains. J. ACM **62**(1), 2:1-2:34 (2015). https://doi.org/10.1145/2629417
2. Agrawal, S., Chatterjee, K., Novotný, P.: Lexicographic ranking supermartingales: an efficient approach to termination of probabilistic programs. Proc. ACM Program. Lang. **2**(POPL), 34:1–34:32 (2018). https://doi.org/10.1145/3158122
3. Akshay, S., Antonopoulos, T., Ouaknine, J., Worrell, J.: Reachability problems for Markov chains. Inf. Process. Lett. **115**(2), 155–158 (2015). https://doi.org/10.1016/j.ipl.2014.08.013
4. Akshay, S., Chatterjee, K., Meggendorfer, T., Đorđe Žikelić: MDPs as distribution transformers: affine invariant synthesis for safety objectives (2023). https://arxiv.org/abs/2305.16796
5. Akshay, S., Genest, B., Vyas, N.: Distribution-based objectives for markov decision processes. In: Dawar, A., Grädel, E. (eds.) Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09–12, 2018, pp. 36–45. ACM (2018). https://doi.org/10.1145/3209108.3209185
6. Alias, C., Darte, A., Feautrier, P., Gonnord, L.: Multi-dimensional rankings, program termination, and complexity bounds of flowchart programs. In: Cousot, R., Martel, M. (eds.) SAS 2010. LNCS, vol. 6337, pp. 117–133. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15769-1_8
7. Alur, R., et al.: Syntax-guided synthesis. In: Irlbeck, M., Peled, D.A., Pretschner, A. (eds.) Dependable Software Systems Engineering, NATO Science for Peace and Security Series, D: Information and Communication Security, vol. 40, pp. 1–25. IOS Press (2015). https://doi.org/10.3233/978-1-61499-495-4-1

8. Asadi, A., Chatterjee, K., Fu, H., Goharshady, A.K., Mahdavi, M.: Polynomial reachability witnesses via stellensätze. In: Freund, S.N., Yahav, E. (eds.) PLDI 2021: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20–25, 2021, pp. 772–787. ACM (2021). https://doi.org/10.1145/3453483.3454076
9. Baier, C., Katoen, J.: Principles of Model Checking. MIT Press, Cambridge (2008)
10. Batz, K., Chen, M., Junges, S., Kaminski, B.L., Katoen, J., Matheja, C.: Probabilistic program verification via inductive synthesis of inductive invariants. In: Sankaranarayanan, S., Sharygina, N. (eds.) TACAS 2023, Part II. LNCS, vol. 13994, pp. 410–429. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-30820-8_25
11. Batz, K., Chen, M., Kaminski, B.L., Katoen, J.-P., Matheja, C., Schröer, P.: Latticed k-induction with an application to probabilistic programs. In: Silva, A., Leino, K.R.M. (eds.) CAV 2021. LNCS, vol. 12760, pp. 524–549. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81688-9_25
12. Beauquier, D., Rabinovich, A.M., Slissenko, A.: A logic of probability with decidable model checking. J. Log. Comput. **16**(4), 461–487 (2006). https://doi.org/10.1093/logcom/exl004
13. Billingsley, P.: Probability and Measure. Wiley, New York (2008)
14. Bradley, A.R., Manna, Z., Sipma, H.B.: Linear ranking with reachability. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 491–504. Springer, Heidelberg (2005). https://doi.org/10.1007/11513988_48
15. Canny, J.F.: Some algebraic and geometric computations in PSPACE. In: Simon, J. (ed.) Proceedings of the 20th Annual ACM Symposium on Theory of Computing, May 2–4, 1988, Chicago, Illinois, USA, pp. 460–467. ACM (1988). https://doi.org/10.1145/62212.62257
16. Carbonneaux, Q., Hoffmann, J., Shao, Z.: Compositional certified resource bounds. In: Grove, D., Blackburn, S.M. (eds.) Proceedings of the 36th ACM SIGPLAN Conference on Programming Language Design and Implementation, Portland, OR, USA, June 15–17, 2015, pp. 467–478. ACM (2015). https://doi.org/10.1145/2737924.2737955
17. Chadha, R., Korthikanti, V.A., Viswanathan, M., Agha, G., Kwon, Y.: Model checking MDPs with a unique compact invariant set of distributions. In: Eighth International Conference on Quantitative Evaluation of Systems, QEST 2011, Aachen, Germany, 5–8 September, 2011, pp. 121–130. IEEE Computer Society (2011). https://doi.org/10.1109/QEST.2011.22
18. Chakarov, A., Sankaranarayanan, S.: Probabilistic program analysis with martingales. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 511–526. Springer (2013). https://doi.org/10.1007/978-3-642-39799-8_34
19. Chatterjee, K., Fu, H., Goharshady, A.K.: Termination analysis of probabilistic programs through Positivstellensatz's. In: Chaudhuri, S., Farzan, A. (eds.) CAV 2016. LNCS, vol. 9779, pp. 3–22. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-41528-4_1
20. Chatterjee, K., Fu, H., Goharshady, A.K., Goharshady, E.K.: Polynomial invariant generation for non-deterministic recursive programs. In: Donaldson, A.F., Torlak, E. (eds.) Proceedings of the 41st ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2020, London, UK, June 15–20, 2020, pp. 672–687. ACM (2020). https://doi.org/10.1145/3385412.3385969

21. Chatterjee, K., Fu, H., Novotný, P., Hasheminezhad, R.: Algorithmic analysis of qualitative and quantitative termination problems for affine probabilistic programs. TOPLAS **40**(2), 7:1–7:45 (2018). https://doi.org/10.1145/3174800
22. Chatterjee, K., Goharshady, A.K., Meggendorfer, T., Zikelic, D.: Sound and complete certificates for quantitative termination analysis of probabilistic programs. In: Shoham, S., Vizel, Y. (eds.) CAV 2022, Part I. LNCS, vol. 13371, pp. 55–78. Springer, Cham (2022). https://doi.org/10.1007/978-3-031-13185-1_4
23. Chatterjee, K., Goharshady, E.K., Novotný, P., Žikelić, Đ.: Proving non-termination by program reversal. In: Freund, S.N., Yahav, E. (eds.) PLDI 2021: 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation, Virtual Event, Canada, June 20–25, 20211, pp. 1033–1048. ACM (2021). https://doi.org/10.1145/3453483.3454093
24. Chatterjee, K., Goharshady, E.K., Novotný, P., Zárevúcky, J., Žikelić, Đ: On lexicographic proof rules for probabilistic termination. In: Huisman, M., Păsăreanu, C., Zhan, N. (eds.) FM 2021. LNCS, vol. 13047, pp. 619–639. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-90870-6_33
25. Chatterjee, K., Novotný, P., Žikelić, Đ.: Stochastic invariants for probabilistic termination. In: POPL, pp. 145–160 (2017). https://doi.org/10.1145/3009837.3009873
26. Cimatti, A., Griggio, A., Schaafsma, B.J., Sebastiani, R.: The MathSAT5 SMT solver. In: Piterman, N., Smolka, S.A. (eds.) TACAS 2013. LNCS, vol. 7795, pp. 93–107. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36742-7_7
27. Colón, M.A., Sankaranarayanan, S., Sipma, H.B.: Linear invariant generation using non-linear constraint solving. In: Hunt, W.A., Somenzi, F. (eds.) CAV 2003. LNCS, vol. 2725, pp. 420–432. Springer, Heidelberg (2003). https://doi.org/10.1007/978-3-540-45069-6_39
28. Colóon, M.A., Sipma, H.B.: Synthesis of linear ranking functions. In: Margaria, T., Yi, W. (eds.) TACAS 2001. LNCS, vol. 2031, pp. 67–81. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45319-9_6
29. Cousot, P., Cousot, R.: Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In: Graham, R.M., Harrison, M.A., Sethi, R. (eds.) Conference Record of the Fourth ACM Symposium on Principles of Programming Languages, Los Angeles, California, USA, January 1977, pp. 238–252. ACM (1977). https://doi.org/10.1145/512950.512973
30. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: The ASTREÉ analyzer. In: Sagiv, M. (ed.) ESOP 2005. LNCS, vol. 3444, pp. 21–30. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31987-0_3
31. Farkas, J.: Theorie der einfachen ungleichungen. Journal für die reine und angewandte Mathematik (Crelles Journal) **1902**(124), 1–27 (1902)
32. Farzan, A., Kincaid, Z.: Compositional recurrence analysis. In: Kaivola, R., Wahl, T. (eds.) Formal Methods in Computer-Aided Design, FMCAD 2015, Austin, Texas, USA, September 27–30, 2015, pp. 57–64. IEEE (2015)
33. Feautrier, P., Gonnord, L.: Accelerated invariant generation for C programs with aspic and c2fsm. In: Delmas, D., Rival, X. (eds.) Proceedings of the Tools for Automatic Program AnalysiS, TAPAS@SAS 2010, Perpignan, France, September 17, 2010. Electronic Notes in Theoretical Computer Science, vol. 267, pp. 3–13. Elsevier (2010). https://doi.org/10.1016/j.entcs.2010.09.014

34. Garg, P., Löding, C., Madhusudan, P., Neider, D.: ICE: a robust framework for learning invariants. In: Biere, A., Bloem, R. (eds.) CAV 2014. LNCS, vol. 8559, pp. 69–87. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-08867-9_5
35. Garg, P., Neider, D., Madhusudan, P., Roth, D.: Learning invariants using decision trees and implication counterexamples. In: Bodík, R., Majumdar, R. (eds.) Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20–22, 2016, pp. 499–512. ACM (2016). https://doi.org/10.1145/2837614.2837664
36. Gario, M., Micheli, A.: Pysmt: a solver-agnostic library for fast prototyping of SMT-based algorithms. In: SMT Workshop, vol. 2015 (2015)
37. Gärtner, B., Matousek, J.: Understanding and using linear programming. Universitext, Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-30717-4
38. Handelman, D.: Representing polynomials by positive linear functions on compact convex Polyhedra. Pacific J. Math. **132**(1), 35–62 (1988)
39. Hoffmann, J., Aehlig, K., Hofmann, M.: Multivariate amortized resource analysis. ACM Trans. Program. Lang. Syst. **34**(3), 14:1–14:62 (2012). https://doi.org/10.1145/2362389.2362393
40. Kaminski, B.L., Katoen, J., Matheja, C., Olmedo, F.: Weakest precondition reasoning for expected runtimes of randomized algorithms. J. ACM 65(5), 30:1–30:68 (2018). https://doi.org/10.1145/3208102
41. Karimov, T., Kelmendi, E., Ouaknine, J., Worrell, J.: What's decidable about discrete linear dynamical systems? In: Raskin, J., Chatterjee, K., Doyen, L., Majumdar, R. (eds.) Principles of Systems Design - Essays Dedicated to Thomas A. Henzinger on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 13660, pp. 21–38. Springer (2022). https://doi.org/10.1007/978-3-031-22337-2_2
42. Kincaid, Z., Breck, J., Boroujeni, A.F., Reps, T.W.: Compositional recurrence analysis revisited. In: Cohen, A., Vechev, M.T. (eds.) Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2017, Barcelona, Spain, June 18–23, 2017, pp. 248–262. ACM (2017). https://doi.org/10.1145/3062341.3062373
43. Kincaid, Z., Cyphert, J., Breck, J., Reps, T.W.: Non-linear reasoning for invariant synthesis. Proc. ACM Program. Lang. **2**(POPL), 54:1–54:33 (2018). https://doi.org/10.1145/3158142
44. Korthikanti, V.A., Viswanathan, M., Agha, G., Kwon, Y.: Reasoning about MDPs as transformers of probability distributions. In: QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15–18 September 2010, pp. 199–208. IEEE Computer Society (2010). https://doi.org/10.1109/QEST.2010.35
45. Kozen, D.: A probabilistic PDL. In: Johnson, D.S., et al. (eds.) Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25–27 April, 1983, Boston, Massachusetts, USA, pp. 291–297. ACM (1983). https://doi.org/10.1145/800061.808758
46. Kretínský, J., Meggendorfer, T.: Conditional value-at-risk for reachability and mean payoff in Markov decision processes. In: Dawar, A., Grädel, E. (eds.) Proceedings of the 33rd Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018, Oxford, UK, July 09–12, 2018, pp. 609–618. ACM (2018). https://doi.org/10.1145/3209108.3209176

47. Kwon, Y., Agha, G.A.: Verifying the evolution of probability distributions governed by a DTMC. IEEE Trans. Software Eng. **37**(1), 126–141 (2011). https://doi.org/10.1109/TSE.2010.80
48. McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science, Springer, Cham (2005). https://doi.org/10.1007/b138392
49. Meggendorfer, T.: Risk-aware stochastic shortest path. In: Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022, pp. 9858–9867. AAAI Press (2022). https://ojs.aaai.org/index.php/AAAI/article/view/21222
50. Meurer, A., et al.: Sympy: symbolic computing in python. PeerJ Comput. Sci. **3**, e103 (2017). https://doi.org/10.7717/peerj-cs.103
51. Mieghem, P.V.: Performance Analysis of Communications Networks and Systems. Cambridge University Press, Cambridge (2006)
52. Morgan, C., McIver, A., Seidel, K.: Probabilistic predicate transformers. ACM Trans. Program. Lang. Syst. **18**(3), 325–353 (1996). https://doi.org/10.1145/229542.229547
53. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78800-3_24
54. Ouaknine, J., Worrell, J.: Decision problems for linear recurrence sequences. In: Finkel, A., Leroux, J., Potapov, I. (eds.) RP 2012. LNCS, vol. 7550, pp. 21–28. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33512-9_3
55. Ouaknine, J., Worrell, J.: Positivity problems for low-order linear recurrence sequences. In: Chekuri, C. (ed.) Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5–7, 2014, pp. 366–379. SIAM (2014). https://doi.org/10.1137/1.9781611973402.27
56. Ouaknine, J., Worrell, J.: On linear recurrence sequences and loop termination. ACM SIGLOG News **2**(2), 4–13 (2015). https://doi.org/10.1145/2766189.2766191
57. Podelski, A., Rybalchenko, A.: A complete method for the synthesis of linear ranking functions. In: Steffen, B., Levi, G. (eds.) VMCAI 2004. LNCS, vol. 2937, pp. 239–251. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24622-0_20
58. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994). https://doi.org/10.1002/9780470316887
59. Putinar, M.: Positive polynomials on compact semi-algebraic sets. Indiana University Math. J. **42**(3), 969–984 (1993)
60. Rodríguez-Carbonell, E., Kapur, D.: Automatic generation of polynomial invariants of bounded degree using abstract interpretation. Sci. Comput. Program. **64**(1), 54–75 (2007). https://doi.org/10.1016/j.scico.2006.03.003
61. Si, X., Dai, H., Raghothaman, M., Naik, M., Song, L.: Learning loop invariants for program verification. In: Bengio, S., Wallach, H.M., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018(December), pp. 3–8, 2018. Montréal, Canada, pp. 7762–7773 (2018). https://proceedings.neurips.cc/paper/2018/hash/65b1e92c585fd4c2159d5f33b5030ff2-Abstract.html

62. Takisaka, T., Oyabu, Y., Urabe, N., Hasuo, I.: Ranking and repulsing supermartingales for reachability in randomized programs. ACM Trans. Program. Lang. Syst. **43**(2), 5:1-5:46 (2021). https://doi.org/10.1145/3450967
63. Wang, P., Fu, H., Goharshady, A.K., Chatterjee, K., Qin, X., Shi, W.: Cost analysis of nondeterministic probabilistic programs. In: McKinley, K.S., Fisher, K. (eds.) Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 2019, Phoenix, AZ, USA, June 22–26, 2019, pp. 204–220. ACM (2019). https://doi.org/10.1145/3314221.3314581
64. Zikelic, D., Chang, B.E., Bolignano, P., Raimondi, F.: Differential cost analysis with simultaneous potentials and anti-potentials. In: Jhala, R., Dillig, I. (eds.) 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation, PLDI 2022, San Diego, CA, USA, 13–17 June 2022, pp. 442–457. ACM (2022). https://doi.org/10.1145/3519939.3523435