Best Probabilistic Transformers

Björn Wachter[1], Lijun Zhang[2,1]

[1] Saarland University, Saarbrücken, Germany
[2] Oxford University Computing Laboratory, UK

Abstract. This paper investigates relative precision and optimality of analyses for concurrent probabilistic systems. Aiming at the problem at the heart of probabilistic model checking – computing the probability of reaching a particular set of states – we leverage the theory of abstract interpretation. With a focus on predicate abstraction, we develop the first abstract-interpretation framework for Markov decision processes which admits to compute both lower and upper bounds on reachability probabilities. Further, we describe how to compute and approximate such abstractions using abstraction refinement and give experimental results.

1 Introduction

Markov decision processes (MDPs) play a crucial role as a semantic model in the analysis of systems with random phenomena like network protocols and randomized algorithms. MDPs feature non-determinism and probabilistic choice. Typically one is interested in computing (maximal or minimal) reachability probabilities, e.g., the probability of delivering three messages after ten transmission attempts. For finite MDPs, probabilistic reachability can be reduced to a linear optimization problem [1]. Recently predicate-abstraction techniques have evolved [2, 3] that scale to realistic programs which map to very large, even infinite MDPs. However, fundamental questions remain open, e.g. for given predicates, what is the most precise abstract program that is still a valid abstraction?

The theory of abstract interpretation [4] has provided answers to such questions in the non-probabilistic case [5] and has served as a foundation and design paradigm for a wide range of other program analyses, e.g. [6–8]. In abstract interpretation, program analyses are expressed in terms of non-standard abstract semantics obtained by replacing the actual domain of computation (also called *concrete domain*) by an *abstract domain*. Concrete and abstract domain are partially ordered sets where ordering describes relative precision of the denotations.

A specification of the *most precise* analysis is given by the composition $f^\sharp = \alpha \circ f \circ \gamma$ of concretization function γ, the functional f characterizing the program semantics and abstraction function α, under the condition that functions α and γ form a Galois connection. Being the limit on the best achievable precision for *any* valid abstraction, functional $f^\sharp$ is called *best transformer* [9].

These concepts are the starting point of our work. Yet a key element is missing: a suitable instantiation of abstract interpretation for our setting.

Related Work. While in [10–13] ideas from abstract interpretation have been applied to probabilistic models, to the best of our knowledge, there is no preceding framework for MDP abstraction with Galois connections and best transformers. Papers [11–13] target deterministic models and not MDPs.

The pioneering contribution in terms of abstract interpretation for MDPs is due to Monniaux [10]. His parametric concept of abstract domains allows to plug in a wealth of base domains from static analysis. However, the resulting abstract domains contain distinct abstract values with the same denotation [14], which means that the abstract domain is not partially ordered (the order is not anti-symmetric). Thus the abstraction precludes Galois connections and is not suitable to develop best transformers.

Further, the aforementioned abstract-interpretation approaches have yet to be combined with abstraction refinement. Refinement admits to adjust the abstraction to the desired precision, which is particularly important in our quantitative setting. In this respect, abstraction refinement based on predicate abstraction has recently shown promising results. The abstraction-refinement method Probabilistic CEGAR [2] computes upper bounds on reachability probabilities for concurrent probabilistic programs, an infinite-state variation of the language of the popular probabilistic model checker `PRISM` [15]. The software model checker in [3] employs predicate abstraction and game-based abstraction [16]. Game-based abstraction maps MDPs to stochastic games. The salient and inspiring point of this influential work is that game-based abstraction yields *both* lower and upper bounds on reachability probabilities, rather than just one bound.

Contribution. Our major theoretical contribution (Sec. 3) is the first abstract-interpretation framework for MDPs which admits to compute both lower and upper bounds on reachability probabilities. This provides a solid basis to reason about the relative precision and optimality of abstract transformers. Further, we prove equivalence of game-based abstraction with best transformers in our framework. Crucial differences to the abstract-interpretation framework [10, 14] are: we consider not only upper but also lower bounds, we target predicate abstraction not classical domains from static analysis, and we express our abstraction in terms of Galois connections.

Our second contribution (Sec. 4) is the first abstraction-refinement technique for concurrent probabilistic programs that yields both lower and upper bounds. Previous analysis techniques for such programs are [17, 2, 18], also based on predicate abstraction. While [2] comes with refinement, it employs an MDP-based abstraction [19] that gives only effective upper bounds. Whereas [18] comes without refinement and uses game-based abstraction, which yields these bounds but requires up to exponentially higher construction cost than MDP-based abstraction and tracking of complex dependencies between commands, which makes all the difference in practice. The basis of our refinement technique is parallel abstraction, a novel abstraction, computable with the same complexity as MDP-based abstraction [17, 2]. Parallel abstraction yields effective lower and upper bounds and combines well with refinement. We have implemented our ideas in the `PASS` tool and report on experimental results (Sec. 5).

2 Background

Sec. 2.1 first recalls basic notions of abstract interpretation including lattices, Galois connections and best transformers. We then introduce the lattice of valuations, the domain for abstract probabilistic reachability analysis, in which probabilities can be represented and computed. In Sec. 2.2, we turn to MDPs and probabilistic reachability.

2.1 Galois connections, Best Transformers and Valuations

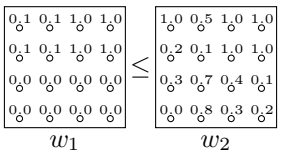
The pair $(A, \leq)$ is a partially-ordered set, or poset, if A is a set and $\leq \subseteq A \times A$ a partial order, i.e. a reflexive, antisymmetric and transitive relation. Let $(A, \leq)$, $(B, \leq)$, $(C, \leq)$ be posets. For two functions $f, g : A \to B$, we write $f \leq g$ if $f(a) \leq g(a)$ for all $a \in A$. We denote the composition of two functions $f_1 : A \to B$ and $f_2 : B \to C$ by $(f_2 \circ f_1) : A \to C$ where $(f_2 \circ f_1)(a) = f_2(f_1(a))$ for all $a \in A$. Function $f : A \to B$ is monotone if for all $a, a' \in A$, $a \leq a' \implies f(a) \leq f(a')$.

A (complete) lattice is a poset $(L, \leq)$ in which each subset $M \subseteq L$ has a greatest lower bound $\bigsqcap M$ and least upper bound $\bigsqcup M$ w.r.t. $\leq$. For a monotone function $f : L \to L$ over a lattice $(L, \leq)$, Tarski's theorem [20] guarantees existence of least and greatest fixpoints, $\mathit{lfp}_{\leq}\ f$ and $\mathit{gfp}_{\leq}\ f$ respectively. They are given by: $\mathit{lfp}_{\leq}(f) = \bigsqcap\{x \in L | f(x) \leq x\}$ and $\mathit{gfp}_{\leq}(f) = \bigsqcup\{x \in L | f(x) \geq x\}$.

In abstract interpretation, the original program semantics, also called concrete semantics, is typically defined over a lattice $(L, \leq)$, called concrete domain, and the abstract semantics is defined over a lattice $(M, \leq)$, called abstract domain. The intuition behind the order $\leq$ in both lattices is that elements higher in the order represent less information. Thus an element $m \in M$ is more precise than another element $m' \in M$ if $m \leq m'$, so m' over-approximates m. For L, the analog holds. The program semantics is described by a concrete transformer, a monotone function $f : L \to L$. An abstract transformer is a monotone function $g^\sharp : M \to M$. Two monotone functions relate abstract and concrete world: the abstraction function $\alpha : L \to M$ and the concretization function $\gamma : M \to L$. The pair (α, γ) is a *Galois connection*, denoted by $(L, \leq) \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} (M, \leq)$, if for all $l \in L$ and $m \in M$, we have $\alpha(l) \leq m \iff l \leq \gamma(m)$.

We call an abstract transformer $g^\sharp : M \to M$ a *valid* abstraction of f if $(f \circ \gamma) \leq (\gamma \circ g^\sharp)$. For transformer $f : L \to L$, the *best transformer* [9], is the composition of functions: $f^\sharp = \alpha \circ f \circ \gamma$. By construction, $f^\sharp$ is the most precise abstract transformer that is a valid abstraction of f, i.e. $f^\sharp \leq g^\sharp$ for any valid transformer $g^\sharp : M \to M$. This follows from properties of the Galois connection.

Lattice of Valuations. A valuation over a set S is a function $w : S \to [0, 1]$ that maps elements of S to probabilities. The valuations $W_S = \{w \mid w : S \to [0, 1]\}$ over S form a lattice $(W_S, \leq)$ with order $\leq$ where $w \leq w'$ if $w(s) \leq w'(s)$ for all $s \in S$. The figure to the right shows two valuations w_1 and w_2 over a set S with 16 elements. Each element is drawn as a circle and the corresponding value is annotated above the circle. We have $w_1 \leq w_2$, i.e., w_2 is an upper bound for w_1, and w_1 a lower bound for w_2. The lattice $(W_S, \geq)$ results by inverting the



w_1 w_2

order in $(W_S, \leq)$. We later use lattice $(W_S, \geq)$ for abstractions yielding lower bounds and lattice $(W_S, \leq)$ for upper bounds. Two lattices are necessary because lattice ordering represents precision, and a lower bound is the more precise the larger it is, while an upper bound is the more precise the smaller it is. To avoid confusion: symbols $\bigsqcap$ and $\bigsqcup$ always refer to the least elements and greatest elements respectively in $(W_S, \leq)$ as given above, and *not* the ones in $(W_S, \geq)$. We have $(\bigsqcap V)(s) = \inf_{w \in V} w(s)$, and $(\bigsqcup V)(s) = \sup_{w \in V} w(s)$ for $V \subseteq W_S$. A *valuation transformer* is a monotone function $f : W_S \to W_S$. Due to duality, we have $(gfp_{\geq}\ f) = (lfp_{\leq}\ f)$ and $(lfp_{\geq}\ f) = (gfp_{\leq}\ f)$.

2.2 Markov decision processes

A *distribution* π over S is a function $\pi : S \to [0, 1]$ such that $\sum_{s \in S} \pi(s) = 1$. Let $Distr_S$ be the set of distributions over S. For a distribution $\pi \in Distr_S$, we denote by $Supp(\pi) = \{s \in S \mid \pi(s) > 0\}$ its support and abbreviate summation over a subset $S' \subseteq S$ by $\pi(S') := \sum_{s \in S'} \pi(s)$.

A *Markov decision process* (MDP) $\mathcal{M}$ is a tuple $(S, I, \mathcal{A}, R)$ where S is a set of states, $I \subseteq S$ is a set of initial states, $\mathcal{A}$ is a finite action alphabet, and $R : S \times \mathcal{A} \rightharpoonup Distr_S$ the transition function. The transition function R is a partial function, as indicated by the $\rightharpoonup$ arrow: only certain actions may be enabled in a state of the MDP or even none. In the latter case, the state is called *absorbing*. For $s \in S$, we denote its enabled actions by $\mathcal{A}(s) = \{a \mid \exists \pi \in Distr_S.\ \pi = R(s, a)\}$, and its out-going distributions by $Distr(s) = \{R(s, a) \mid a \in \mathcal{A}(s)\}$. For $a \in \mathcal{A}(s)$, we define $\pi_{(s,a)} := R(s, a)$ and say that $(s, a, \pi_{(s,a)})$ is a transition of $\mathcal{M}$.

A path is a sequence $(s_0, a_0, \pi_0), (s_1, a_1, \pi_1), \ldots$ such that $s_0 \in I$, (s_i, a_i, π_i) are transitions of $\mathcal{M}$, and $s_{i+1} \in Supp(\pi_i)$ for all $i \in \mathbb{N}$. Let $Path(\mathcal{M})$ denote the set of all paths over $\mathcal{M}$. Similarly finite paths can be defined. For $\beta \in Path(\mathcal{M})$, let $\beta[i] = s_i$ denote the $(i + 1)$-th state of β.

Markov chains are special cases of MDPs, deterministic MDPs where for every state s there is at most one enabled transition $|\mathcal{A}(s)| \leq 1$. Unlike a Markov chain, an MDP is not a fully determined stochastic process. In order to obtain a probability measure, the notion of a *strategy* is needed to resolve nondeterminism. In general, a strategy σ of an MDP $\mathcal{M}$ is a function from finite paths to distributions over actions. We denote the set of strategies of $\mathcal{M}$ by $\Sigma_{\mathcal{M}}$. For a given state $s \in S$ and a strategy σ, let Pr_s^σ denote the corresponding probability measure [1] over $Path(\mathcal{M})$.

Probabilistic Reachability. Let $\mathcal{M} = (S, I, \mathcal{A}, R)$ be an MDP, and let $F \subseteq S$ be a set of goal states. Let $p_s^\sigma(F) := Pr_s^\sigma(\{\beta \in Path(\mathcal{M}) \mid \exists i \in \mathbb{N} :\ \beta[i] \in F\})$ denote the probability of reaching a goal state in F from state $s \in S$ with strategy σ. For a fixed strategy σ, this defines a valuation $p^\sigma(F) \in W_S$ which maps a state s to $p_s^\sigma(F)$. In the context of MDPs, one studies minimal $p^-(F) \in W_S$ and maximal $p^+(F) \in W_S$ reachability probabilities where $p^-(F) = \bigsqcap\{p^\sigma(F) \mid \sigma \in \Sigma_{\mathcal{M}}\}$ is the infimum and $p^+(F) = \bigsqcup\{p^\sigma(F) \mid \sigma \in \Sigma_{\mathcal{M}}\}$ the supremum over all strategies. Below we define two valuation transformers to characterize the minimal and maximal reachability probabilities.

Definition 1 (Valuation Transformers for MDPs). *Let $F_0 \subseteq S$ be the set of states that cannot reach states in F. The valuation transformer $pre_F^- : W_S \to W_S$ is defined by: $pre_F^-(w)(s) = 1$ if $s \in F$, $pre_F^-(w)(s) = 0$ if $s \in F_0$, and otherwise:*

$$pre_F^-(w)(s) = \min_{a \in \mathcal{A}(s)} \sum_{s' \in S} \pi_{(s,a)}(s') \cdot w(s').$$

The valuation transformer $pre_F^+ : W_S \to W_S$ is defined analogously, with the difference that it maximizes over all enabled actions.

Example 1. We illustrate the transformer pre_F^- by considering the MDP in Figure 1. Assume that the goal states are given by the set $F = \{s_2, s_3\}$ and that valuation w assigns probability 1 to s_0, s_2 and s_3 respectively and probability 0 to all other states. Inserting the values and solving for state s_0, we get $pre_F^-(w)(s_0) = \min\{w(s_0), \frac{w(s_1)+w(s_2)}{2}, \frac{w(s_2)+w(s_3)+w(s_4)}{3}\} = \min\{1, \frac{1}{2}, \frac{2}{3}\} = \frac{1}{2}$. For state s_4, we have: $pre_F^-(w)(s_4) = \frac{1}{3}w(s_0) + \frac{2}{3}w(s_4) = \frac{1}{3}$. ■

Minimal and maximal reachability probabilities are expressible as *least fixpoints* of valuation transformers pre_F^- and pre_F^+ respectively [21] or more formally $p^-(F) = lfp_{\leq}\ pre_F^-$ and $p^+(F) = lfp_{\leq}\ pre_F^+$. This fixpoint characterization together with the background on abstract interpretation allows us to express abstractions for probabilistic reachability.

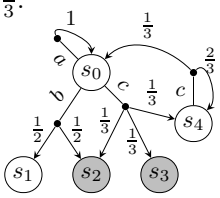


Fig. 1.

3 Abstraction

We present our novel abstract-interpretation framework for MDP abstraction. The concrete and abstract domain are given by lattices of valuations. Thereby lattice order expresses the concept of lower and upper bounds. In Sec. 3.1, we develop the abstract domain and apply the resulting abstraction framework to transformers, their fixpoints and particularly probabilistic reachability. Sect. 3.3 reveals a strong connection between the game-theoretical construction of game-based abstraction and certain best transformers.

3.1 Lower- and Upper-Bound Abstraction

Let S be a set of states. A partition Q of S is a finite set of pairwise disjoint, nonempty subsets of S such that $S = \bigcup_{B \in Q} B$. Elements of Q are called blocks. For a state $s \in S$, we denote by $\overline{s}$ the unique block B containing s, i.e., $s \in B$. Abstract valuations are valuations over blocks, elements of W_Q.

We give two abstraction functions that, given a valuation over states, yield a valuation over blocks: *lower-bound abstraction* $\alpha^{\mathsf{l}} : W_S \to W_Q$ returns the infimum of the values $\alpha^{\mathsf{l}}(w)(B) = \inf_{s \in B} w(s)$ within a block while *upper-bound abstraction* $\alpha^{\mathsf{u}} : W_S \to W_Q$ returns the supremum $\alpha^{\mathsf{u}}(w)(B) = \sup_{s \in B} w(s)$.

The *concretization* of an abstract valuation $w^\sharp \in W_Q$ is the valuation over states $\gamma(w^\sharp)$ that assigns each state the value of its block $\gamma(w^\sharp)(s) = w^\sharp(\overline{s})$. This defines the concretization function $\gamma : W_Q \to W_S$.

A lower bound is the more precise the larger it is, while the converse is true for an upper bound. This notion of precision is reflected by the lattice order: the order is $\geq$ if we compare lower bounds, and $\leq$ for upper bounds.

We obtain two Galois connections corresponding to α^{l} and α^{u} respectively:

Lemma 1 (Galois Connections). *For a given partition Q, let α^{l}, α^{u}, γ be the functions defined above. We have the following two Galois connections:*

(a) $(W_S, \geq) \underset{\alpha^{\mathsf{l}}}{\overset{\gamma}{\leftrightarrows}} (W_Q, \geq)$ *(lower-bound abstraction)*

(b) $(W_S, \leq) \underset{\alpha^{\mathsf{u}}}{\overset{\gamma}{\leftrightarrows}} (W_Q, \leq)$ *(upper-bound abstraction)*

Proof. We focus on lower-bound abstraction. Monotonicity of α^{l} and γ follows directly by definition. It remains to show that for all $w \in W_S$ and $w^\sharp \in W_Q$, it holds that: $w \geq \gamma(w^\sharp) \Leftrightarrow \alpha^{\mathsf{l}}(w) \geq w^\sharp$. First assume $w \geq \gamma(w^\sharp)$. For $B \in Q$, we have $\alpha^{\mathsf{l}}(w)(B) = \inf_{s \in B} w(s) \geq \inf_{s \in B} \gamma(w^\sharp)(s) = w^\sharp(B)$. Now assume $\alpha^{\mathsf{l}}(w) \geq w^\sharp$. Then, for all $s \in S$, we have $w(s) \geq \alpha^{\mathsf{l}}(w)(\overline{s}) \geq w^\sharp(\overline{s}) = \gamma(w^\sharp)(s)$. The proof for upper-bound abstraction works analogously. ∎

The two Galois connections are illustrated in Figure 2. The big dashed box on the left represents valuations over states W_S (concrete domain), the one on the right represents valuations over blocks W_Q (abstract domain). The partition into blocks B_1, B_2, B_3 is depicted by rectangles surrounding states. Consider the valuation w with the thick border. Abstraction $\alpha^{\mathsf{l}}(w)$ provides a lower bound, i.e. $\gamma(\alpha^{\mathsf{l}}(w)) \leq w$. Taking the α^{u}-abstraction yields an upper bound $\alpha^{\mathsf{u}}(w)$, i.e. we get $w \leq \gamma(\alpha^{\mathsf{u}}(w))$.

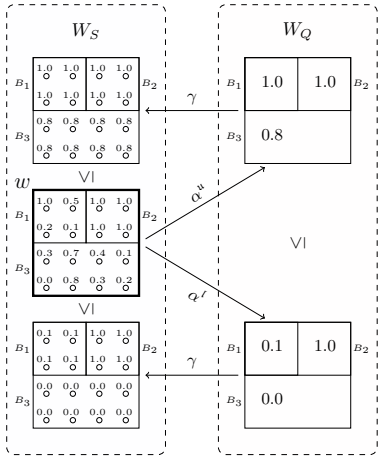


Fig. 2. Functions α^{l} and α^{u}.

Remark. We point out crucial differences to Monniaux's framework [10]: unlike in this paper, only upper bounds *not* lower bounds are computed, his abstract domains are in general not partially ordered, and the concrete domain consists of sets of valuations rather than valuations. Therefore, unlike in our setting, the concretization function maps an abstract valuation to a set of valuations, as opposed to a single valuation.

Lower and Upper Bounds of Fixpoints. Aiming for probabilistic reachability, we consider fixpoints of valuation transformers. If a valuation transformer is bounded from below and above by two abstract transformers which are valid abstractions w.r.t. lower-bound and upper-bound abstraction respectively, the fixpoints of these two abstract transformers enclose the fixpoint of the valuation transformer, which is formalized in Lemma 2. In the following, we fix Q as the given partition, and let α^{l}, α^{u}, γ be the functions defined above.

Lemma 2 (Bounds from Valid Transformers). *Let $f : W_S \to W_S$. Let $f_1^\sharp, f_2^\sharp : W_Q \to W_Q$ be valuation transformers such that $f_1^\sharp$ is a valid lower-bound of f and $f_2^\sharp$ a valid upper-bound abstraction of f, i.e., $\gamma \circ f_1^\sharp \leq f \circ \gamma \leq \gamma \circ f_2^\sharp$. Then the following inequality holds regarding the fixpoints of these functions:*

$$\gamma\left(gfp_{\geq}(f_1^\sharp)\right) \quad \leq \quad lfp_{\leq}(f) \quad \leq \quad \gamma\left(lfp_{\leq}(f_2^\sharp)\right) \ .$$

Proof. Let $w^* = lfp_{\leq}(f_2^\sharp)$ be the least fixpoint of $f_2^\sharp$. It holds that $f_2^\sharp(w^*) = w^*$ and hence $(\gamma \circ f_2^\sharp)(w^*) = \gamma(w^*)$. By assumption, we have $f \circ \gamma \leq \gamma \circ f_2^\sharp$, which implies $(f \circ \gamma)(w^*) \leq (\gamma \circ f_2^\sharp)(w^*) = \gamma(w^*)$ and $\gamma(w^*) \in \{x \in W_S \mid f(x) \leq x\}$. Hence $lfp_{\leq}(f) = \bigsqcap\{x \in W_S \mid f(x) \leq x\} \leq \gamma(w^*)$, as claimed. The other inequality can be shown in a dual way. ■

For any concrete valuation transformer f, the fixpoints of the best transformers w.r.t. lower- and upper-bound abstraction enclose the least fixpoint of f:

Lemma 3 (Fixpoints of Best Transformers). *Let $f : W_S \to W_S$ be a given valuation transformer. Then the following inequalities hold:*

$$\gamma\left(gfp_{\geq}(\alpha^{\mathsf{l}} \circ f \circ \gamma)\right) \quad \leq \quad lfp_{\leq}(f) \quad \leq \quad \gamma\left(lfp_{\leq}(\alpha^{\mathsf{u}} \circ f \circ \gamma)\right) \ .$$

The proof follows immediately by applying Lemma 2 to the best transformers $f_1^\sharp = (\alpha^{\mathsf{l}} \circ f \circ \gamma)$ and $f_2^\sharp = (\alpha^{\mathsf{u}} \circ f \circ \gamma)$. For probabilistic reachability, we consider four different best transformers: $\alpha^{\{\mathsf{l},\mathsf{u}\}} \circ pre_F^{\{-,+\}} \circ \gamma$ where the abstraction function controls whether we get lower or upper bounds, and the valuation transformer controls whether maximal or minimal reachability probability is considered. Exploiting the fact that $p^-(F) = lfp_{\leq}\ pre_F^-$ and $p^+(F) = lfp_{\leq}\ pre_F^+$, we have the connection to probabilistic reachability:

Theorem 1 (Bounds for Probabilistic Reachability). *Let $\mathcal{M} = (S, I, \mathcal{A}, R)$ be an MDP and let $F \subseteq S$ be a set of goal states. Then we have:*

$$\gamma(gfp_{\geq}(\alpha^{\mathsf{l}} \circ pre_F^+ \circ \gamma)) \leq p^+(F) \leq \gamma(lfp_{\leq}(\alpha^{\mathsf{u}} \circ pre_F^+ \circ \gamma))$$
$$\gamma(gfp_{\geq}(\alpha^{\mathsf{l}} \circ pre_F^- \circ \gamma)) \leq p^-(F) \leq \gamma(lfp_{\leq}(\alpha^{\mathsf{u}} \circ pre_F^- \circ \gamma))$$

The best transformer $(\alpha^{\mathsf{u}} \circ pre_F^- \circ \gamma)$ for the upper bound of minimal reachability contains an alternation between minimization, as in pre_F^-, and maximization, as in α^{u}. This suggests a connection to stochastic games where minimization and maximization are the objectives of two adversarial players. After an interlude on stochastic games in Sec. 3.2, we make the connection to game-based abstraction.

3.2 Stochastic games

We consider turn-based stochastic games with two players [22]. A stochastic game is a tuple $\mathcal{G} = ((V, E), V_{init}, (V_1, V_2, V_p), \delta)$ where (V, E) is a *finite* directed graph with edges $E \subseteq V \times V$, $V_{init} \subseteq V_1$ is the set of initial vertices, (V_1, V_2, V_p) is a partition of the set V, and $\delta : V_p \to Distr_V$ where $\delta(v)(v') > 0$ implies that $(v, v') \in E$. The vertex sets V_1, V_2 are called player 1 vertices and player 2

vertices respectively. For $v \in V$, let $E(v) = \{w \mid (v,w) \in E\}$ be the successors of v. A *play* of the game is a sequence $\omega = v_0 v_1 \ldots$ such that $v_{i+1} \in E(v_i)$ for all $i \in \mathbb{N}$. Let $\omega[i] = v_i$ denote the $(i+1)$-th vertex of ω, and denote the last vertex by $last(\omega) = v_n$ if ω is finite.

A player 1 strategy is a function $\sigma_1 : V^* V_1 \to Distr_V$ such that for any finite play ω, $\sigma_1(\omega)(v) > 0$ implies that $(last(\omega), v) \in E$. Player 2 strategies are defined analogously. A strategy σ_i is called *pure memoryless* if it does not use randomization and is memoryless: it is a function $\sigma_i : V_i \to V$ for $i = 1, 2$. For any vertex $v \in V_{init}$, a fixed pair of strategies corresponds probability measure $Pr_v^{\sigma_1,\sigma_2}$ over infinite plays. Given a vertex v, a *reachability objective* $F \subseteq V_1$, and strategies σ_1, σ_2, $p_v^{\sigma_1,\sigma_2}(F)$ denotes the probability of reaching F starting in v: $p_v^{\sigma_1,\sigma_2}(F) = Pr_v^{\sigma_1,\sigma_2}(\{\omega \mid \exists i \in \mathbb{N} : \ \omega[i] \in F\})$. This defines a valuation $p^{\sigma_1,\sigma_2}(F) \in W_V$. *Optimal valuations* for player 1 and player 2 w.r.t. F are defined by: $\sup_{\sigma_1} \inf_{\sigma_2} p^{\sigma_1,\sigma_2}(F), \inf_{\sigma_1} \sup_{\sigma_2} p^{\sigma_1,\sigma_2}(F) \in W_V$ respectively. Player 1 strategy σ_1 is *optimal* for $v \in V$ if $\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) = \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F)$. The optimal player 2 strategy can be defined similarly. We also consider the cases where both players cooperate $\inf_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F)$ and $\sup_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F)$. Below we define four valuation transformers to characterize these optimal valuations:

Definition 2 (Valuation Transformers for Games). *Given a reachability objective F, let $F_0 \subseteq V_1$ be the set of vertices that cannot reach F. The valuation transformer $pre_F^{+-} : W_{V_1} \to W_{V_1}$ is defined by: $pre_F^{+-}(d)(v)$ equals 1 if $v \in F$ and 0 if $v \in F_0$, and otherwise,*

$$pre_F^{+-}(d)(v) = \max_{v_2 \in E(v)} \min_{v_p \in E(v_2)} \sum_{v' \in E(v_p)} \delta(v_p)(v') \cdot d(v').$$

The valuation transformers $pre_F^{--}, pre_F^{-+}, pre_F^{++}$ can be defined analogously by changing the extrema in the summation accordingly, e.g. pre_F^{-+} minimizes over $E(v)$ and maximizes over $E(v_2)$.

The optimal valuations are least fixpoints of valuation transformers, e.g. $\sup_{\sigma_1} \inf_{\sigma_2} p^{\sigma_1,\sigma_2}(F) = lfp_{\leq}\ pre_F^{+-}$, and $\sup_{\sigma_1} \sup_{\sigma_2} p^{\sigma_1,\sigma_2}(F) = lfp_{\leq}\ pre_F^{++}$.

3.3 Best Transformers and Game-Based Abstraction

Given a finite partition, game-based abstraction maps an MDP to a stochastic game[1]. The blocks of the partition are the player 1 vertices and player 2 vertices are sets of abstract distributions. The abstraction of a distribution $\pi \in Distr_S$ is the distribution $\overline{\pi} \in Distr_Q$ with $\overline{\pi}(B) := \sum_{s \in B} \pi(s)$. The abstraction of a set of distributions D is the set $\overline{D} = \{\overline{\pi} \mid \pi \in D\}$ for $D \subseteq Distr_S$.

[1] In full generality, infinite stochastic games may arise through game-based abstraction. From now on, we assume finiteness. In effect, this excludes MDPs with infinitely many different transition probabilities, which are not representable in our modeling language. In subsequent proofs of this section, finiteness of the games also implies that the abstraction functions α^{u} and α^{l} are applied to valuations for which not only infima and suprema but minima and maxima exist.

Definition 3 (Game-based Abstraction [16]). *Let $\mathcal{M} = (S, I, \mathcal{A}, R)$ be an MDP, and Q be a partition. The game abstraction of $\mathcal{M}$ w.r.t. Q is the stochastic game $\mathcal{G}_{\mathcal{M},Q} = ((V, E), V_{init}, (V_1, V_2, V_p), \delta)$ where the player 1 vertices $V_1 = Q$ are given by the blocks, the player 2 vertices $V_2 = \{\overline{Distr(s)} \mid s \in S\} \subseteq 2^{Distr_Q}$ are sets of distributions and the vertices $V_p = \{\overline{\pi} \mid \exists s \in S : \ \pi \in Distr(s)\}$ distributions. $\delta : V_p \to Distr_V$ is the identity function. The initial vertices are $V_{init} = \{B \in Q \mid B \cap I \neq \emptyset\}$ and the edges E are given by:*

$$E = \{(v_1, v_2) \mid v_1 \in V_1, \exists s \in v_1 : \ v_2 = \overline{Distr(s)}\}$$
$$\cup \{(v_2, v_p) \mid v_2 \in V_2, v_p \in v_2\} \cup \{(v_p, v_1) \mid v_p \in V_p, v_p(v_1) > 0\} \ .$$

Intuitively, a player 1 decision is a concretization step for a given block. The abstract distributions of a player 2 vertex correspond to the out-going distributions of a concrete state. The player 2 decision is then like the application of the concrete transformer ensued by an abstraction step. In fact, the best transformers w.r.t. lower and upper bound abstraction (see Sec. 3.1) are exactly the valuation transformers of game-based abstraction:

Theorem 2 (Game-based Abstraction and Best Transformer). *Let $\mathcal{M}$ be an MDP and $F \subseteq S$ a set of goal states. Further, consider the partition Q of S such that the goal states F in $\mathcal{M}$ are exactly representable: i.e., $F = \bigcup_{B \in F^\sharp} B$ for a suitable $F^\sharp \subseteq Q$. Let $pre^{\pm\pm}_{F^\sharp}$ be the valuation transformers in the game $\mathcal{G}_{\mathcal{M},Q}$ as defined in Def. 2. Then it holds that:*

$$pre^{--}_{F^\sharp} = \alpha^{\mathfrak{l}} \circ pre^{-}_{F} \circ \gamma \quad , \quad pre^{+-}_{F^\sharp} = \alpha^{\mathfrak{u}} \circ pre^{-}_{F} \circ \gamma$$
$$pre^{-+}_{F^\sharp} = \alpha^{\mathfrak{l}} \circ pre^{+}_{F} \circ \gamma \quad , \quad pre^{++}_{F^\sharp} = \alpha^{\mathfrak{u}} \circ pre^{+}_{F} \circ \gamma$$

Proof. We sketch the proof for $pre^{-+}_{F^\sharp} = \alpha^{\mathfrak{l}} \circ pre^{+}_{F} \circ \gamma$, i.e., the claim is that, for all $w^\sharp \in W_Q$ and $v \in V_1$, $pre^{-+}_{F}(w^\sharp)(v) = \left((\alpha^{\mathfrak{l}} \circ pre^{+} \circ \gamma)(w^\sharp)\right)(v)$. The claim is trivially fulfilled if $v \in F^\sharp \cup F^\sharp_0$. Otherwise the transformer of the game is defined as $pre^{-+}_{F^\sharp}(w^\sharp)(v) = \min_{v_2 \in E(v)} \max_{v_p \in v_2} \sum_{v' \in V_1} \delta(v_p)(v') \cdot w^\sharp(v')$. The successors of vertex $v \in V_1$ are given by $E(v) = \{\overline{Distr(s)} \mid s \in v\}$. It is easy to see that $pre^{-+}_{F^\sharp}(w^\sharp)(v) = \min_{s \in v} \max_{\pi^\sharp \in \overline{Distr(s)}} \sum_{v' \in V_1} \pi^\sharp(v') \cdot w^\sharp(v')$. Observe that for a distribution π and a block $v' \in V_1$, we have by definition $\overline{\pi}(v') = \sum_{s \in v'} \pi(s)$ and thus $\sum_{v' \in V_1} \overline{\pi}(v') \cdot w^\sharp(v') = \sum_{s' \in S} \pi(s') \cdot w^\sharp(\overline{s'})$. As a final step, we get the equality $\left((\alpha^{\mathfrak{l}} \circ pre^{+}_{F} \circ \gamma)(w^\sharp)\right)(v) = \min_{s \in v} \max_{\pi \in Distr(s)} \sum_{s' \in S} \pi(s') \cdot w^\sharp(\overline{s'})$, which proves the claim. ∎

As corollary of Theorem 2, one obtains that the valuation transformers of the games are a valid abstraction for minimal and maximal reachability. Together with Theorem 1 this proves that game-based abstraction yields lower and upper bounds on probabilistic reachability:

$$\gamma(\inf_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F^\sharp)) \leq p^-(F) \leq \gamma(\sup_{\sigma_1} \inf_{\sigma_2} p^{\sigma_1,\sigma_2}(F^\sharp)) \tag{1}$$

$$\gamma(\inf_{\sigma_1} \sup_{\sigma_2} p^{\sigma_1,\sigma_2}(F^\sharp)) \leq p^+(F) \leq \gamma(\sup_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F^\sharp)) \tag{2}$$

Theorem 2 establishes that the obtained probability bounds are optimal, i.e. any valid abstraction cannot yield more precise bounds.

One can thus compute best transformers by game-based abstraction, yet computational cost is higher than for abstractions that map to MDPs [18]. In the next section, we introduce abstractions for concurrent programs that alleviate this problem and still yield effective lower and upper bounds.

4 Abstraction Refinement for Concurrent Programs

We discuss concurrent probabilistic programs in Sec. 4.1. In Sec. 4.2, we present a novel abstraction tailored to these programs and introduce the corresponding game construction in Sec. 4.3. Together with the refinement algorithm in Sec. 4.4, we obtain the first abstract-refinement method for infinite-state concurrent probabilistic programs that provides both lower and upper bounds.

4.1 Concurrent Probabilistic Programs.

As in [2], we consider a variation of the popular PRISM language [15] that additionally supports integer and real variables. We now give the abstract syntax and the semantics of concurrent probabilistic programs. We fix a finite set of program variables $\mathtt{X}$ and a finite set of actions $\mathcal{A}$. We denote the expressions over the variables V by $Expr_V$ and Boolean expressions by $BExpr_V$. An *assignment* is a function $E : \mathtt{X} \to Expr_{\mathtt{X}}$.

A *program* $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ consists of an initial condition $\mathtt{I} \in BExpr_{\mathtt{X}}$ and commands $\mathtt{C}$. A *command* $\mathtt{c}$ consists of a unique action a, a guard $\mathtt{g} \in BExpr_{\mathtt{X}}$ and assignments $E_{\mathtt{u}_1}, ..., E_{\mathtt{u}_k}$ weighted with probabilities $p_1, ..., p_k$ where $\sum_{i=1}^{k} p_i = 1$. We denote by $\mathtt{X}' = E$ the simultaneous update E of variables $\mathtt{X}$.

```
module two_chains
m : [0..3];      // control flow
x : int;         // counter variable
[a] m=0 -> 1.0: (x'=1000) & (m'=1);
[b] m=0 -> 1.0: (x'=2) & (m'=1);
[c] m=1 & x>0 -> 0.3: (x'=x-1) + 0.7: (m'=3);
[d] m=1 & x<=0 -> 1.0: (m'=2);
endmodule
init
 m = 0 & x = 0
endinit
```

Fig. 3. Example program with variables `m` and `x` and four commands.

With the i-th update of $\mathtt{c}$, we associate a unique update label $\mathtt{u}_i \in \mathtt{U}$. Updates are separated by a "+": $[a]\ \mathtt{g} \to \mathtt{p}_1 : \mathtt{X}'{=}E_{\mathtt{u}_1}\ \mathtt{+}\ \ldots\ \mathtt{+}\ \mathtt{p}_k : \mathtt{X}'{=}E_{\mathtt{u}_k}$. If the guard is satisfied, the i-th update executes with probability p_i. For a command $\mathtt{c}$, we write $a_{\mathtt{c}}$ for its action, $\mathtt{g}_{\mathtt{c}}$ for its guard and omit subscripts if the command is clear from context.

A *state* over variables $\mathtt{X}$ is a type-consistent total function from variables in $\mathtt{X}$ to their semantic domains. We denote the set of states by $S(\mathtt{X})$, or S for short, and a single state by s. For an expression $\mathtt{e} \in Expr_{\mathtt{X}}$, we denote by $[\![\mathtt{e}]\!]_s$ its valuation in state s. For a Boolean expression $\mathtt{e} \in BExpr_{\mathtt{X}}$, we have $[\![\mathtt{e}]\!]_s \in \{0, 1\}$ and denote by $[\![\mathtt{e}]\!] = \{s \in S \mid [\![\mathtt{e}]\!]_s = 1\}$ the set of states that fulfill $\mathtt{e}$.

The semantics of a program $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ is the MDP $\mathcal{M} = (S, I, \mathcal{A}, R)$ with states $S = S(\mathtt{X})$, initial states $I = [\![\mathtt{I}]\!]$, actions $\mathcal{A} = \{a_{\mathtt{c}} \mid \mathtt{c} \in \mathtt{C}\}$, and transitions induced by the commands. Consider $s \in S$ and $a_{\mathtt{c}} \in \mathcal{A}$. If $s \in [\![\mathtt{g}_{\mathtt{c}}]\!]$, we define $R(s, a_{\mathtt{c}}) = \pi$ such that π fulfills the following dependency where $\{|\ldots|\}$ delimits

a multiset: $\pi(s') = \sum_{i=1}^{k} \{|p_i \mid \forall \mathtt{x} \in \mathtt{X} :\ s'(\mathtt{x}) = [\![E_{\mathtt{u}_i}(\mathtt{x})]\!]_s|\}$. We use a multiset since two updates may have the same probability and yield the same state.

4.2 Parallel Abstraction

A concurrent probabilistic program consists of the parallel composition of commands. In parallel abstraction, abstract transformers for the program are obtained by the parallel composition of the abstract transformers of the commands.

We first focus on abstraction for maximal probabilistic reachability. Before defining the abstract transformers, we reformulate concrete transformer in terms of transformers $pre_F^+[a]$ for the individual actions. Let $w \in W_S$ be a valuation and $s \in S$ a state. Then $pre[a]_F^+(w)(s)$ equals 1 if s is a goal state, 0 if s cannot reach a goal state or action a is not enabled on s, and, lastly, $\sum_{s' \in S} \pi_{(s,a)}(s') \cdot w(s')$ otherwise. It is obvious that the transformer pre_F^+ for maximal reachability of goal states F is given by:

$$pre_F^+(w)(s) = \max_{a \in \mathcal{A}(s)} pre[a]_F^+(w)(s) \ . \tag{3}$$

We assume that the partition Q is chosen such that the goal states can be represented precisely, i.e. there exists a set of blocks $F^\sharp \subseteq Q$ with $F = \bigcup_{B \in F^\sharp} B$. Further, we assume that, without loss of generality, a block in a partition contains either only absorbing or no absorbing states.

For a block $B \in Q$, we denote by $\mathcal{A}(B)$ the set $\{a \in \mathcal{A} \mid \exists s \in B :\ a \in \mathcal{A}(s)\}$ of actions that are enabled some state in B. By combining abstract transformers of the commands, we get the abstract transformer for the whole program.

Definition 4 (Maximal Parallel Abstraction). *We define the respective abstract transformers for the lower and upper bounds of maximal reachability:*

$$\widetilde{pre}_{F^\sharp}^{\mathsf{l}+}(w^\sharp)(B) := \max_{a \in \mathcal{A}(B)} (\alpha^{\mathsf{l}} \circ (pre[a]_F^+) \circ \gamma)(w^\sharp)(B) \ ,$$

$$\widetilde{pre}_{F^\sharp}^{\mathsf{u}+}(w^\sharp)(B) := \max_{a \in \mathcal{A}(B)} (\alpha^{\mathsf{u}} \circ (pre[a]_F^+) \circ \gamma)(w^\sharp)(B) \ .$$

Similar to maximal reachability, we define transformer $pre[a]_F^- : W_S \to W_S$ for action a as follows. For a valuation $w \in W_S$ and state s, $pre[a]_F^-(w)(s)$ equals 1 if state s is a goal state or the action is not enabled, 0 if the goal states are not reachable from s, and $\sum_{s' \in S} \pi_{(s,a)}(s') \cdot w(s')$ otherwise. As for maximal reachability, the transformer pre_F^- is given by:

$$pre_F^-(w)(s) = \min_{a \in \mathcal{A}(s)} pre[a]_F^-(w)(s) \ . \tag{4}$$

Now we define the abstract transformers for the two bounds in terms of the best transformers of individual commands.

Definition 5 (Minimal Parallel Abstraction). *We define the respective abstract transformers for the lower and upper bounds of minimal reachability:*

$$\widetilde{pre}_{F^\sharp}^{\mathsf{l}-}(w^\sharp)(B) := \min_{a \in \mathcal{A}(B)} (\alpha^{\mathsf{l}} \circ (pre[a]_F^-) \circ \gamma)(w^\sharp)(B) \ ,$$

$$\widetilde{pre}_{F^\sharp}^{\mathsf{u}-}(w^\sharp)(B) := \min_{a \in \mathcal{A}(B)} (\alpha^{\mathsf{u}} \circ (pre[a]_F^-) \circ \gamma)(w^\sharp)(B) \ .$$

The following theorem states that the introduced maximal and minimal abstract transformers are valid abstractions. For $\widetilde{pre}^{\mathsf{u}+}_{F^\sharp}$ and $\widetilde{pre}^{\mathsf{l}-}_{F^\sharp}$ one can even show a stronger result: these transformers are exactly the best transformers $\alpha^{\mathsf{u}} \circ pre^{+}_{F} \circ \gamma$ and $\alpha^{\mathsf{l}} \circ pre^{-}_{F} \circ \gamma$ respectively. In general, this does not hold for the other two transformers. Overall we have:

Theorem 3 (Validity of Parallel Abstraction).

1. $\widetilde{pre}^{\mathsf{u}+}_{F^\sharp}$ *equals the best transformer* $\alpha^{\mathsf{u}} \circ pre^{+}_{F} \circ \gamma$,
2. $\widetilde{pre}^{\mathsf{l}+}_{F^\sharp}$ *is a valid abstraction of* $pre^{+}_{F^\sharp}$ *w.r.t. the Galois connection* $(\alpha^{\mathsf{l}}, \gamma)$, *i.e.,* $\gamma \circ \widetilde{pre}^{\mathsf{l}+}_{F^\sharp} \leq pre^{+}_{F} \circ \gamma$,
3. $\widetilde{pre}^{\mathsf{u}-}_{F^\sharp}$ *is a valid abstraction of* $pre^{-}_{F^\sharp}$ *w.r.t. the Galois connection* $(\alpha^{\mathsf{u}}, \gamma)$, *i.e.,* $pre^{-}_{F} \circ \gamma \leq \gamma \circ \widetilde{pre}^{\mathsf{u}-}_{F^\sharp}$,
4. $\widetilde{pre}^{\mathsf{l}-}_{F^\sharp}$ *equals the best transformer* $\alpha^{\mathsf{l}} \circ pre^{-}_{F} \circ \gamma$.

Proof. Part (1): Let $w^\sharp \in W_Q, B \in Q \setminus F^\sharp$. By definition, we have the equality $(\alpha^{\mathsf{u}} \circ pre^{+}_{F} \circ \gamma)(w^\sharp)(B) = \max_{s \in B} \max_{\pi \in Distr(s)} \sum_{s' \in S} \pi(s') \cdot w^\sharp(\overline{s'})$. Moreover, we have $\widetilde{pre}^{\mathsf{u}+}_{F^\sharp}(w^\sharp)(B) = \max_{a \in \mathcal{A}(B)} (\alpha^{\mathsf{u}} \circ (pre[a]^{+}_{F}) \circ \gamma)(w^\sharp)(B)$ by Def. 4. This can be rewritten to: $\widetilde{pre}^{\mathsf{u}+}_{F^\sharp}(w^\sharp)(B) = \max_{s \in B} \max_{a \in \mathcal{A}(s)} \sum_{s' \in S} \pi_{(s,a)}(s') \cdot w^\sharp(\overline{s'})$, which is the same as $\max_{s \in B} \max_{\pi \in Distr(s)} \sum_{s' \in S} \pi(s') \cdot w^\sharp(\overline{s'})$. We are done.

Part (3): we show $(pre^{-}_{F} \circ \gamma)(w^\sharp)(B) \leq (\gamma \circ \widetilde{pre}^{\mathsf{u}-}_{F^\sharp})(w^\sharp)(B)$ for all $w^\sharp \in W_Q$ and $B \in Q$. We consider the Galois connection $(\alpha^{\mathsf{u}}, \gamma)$ with order $\leq$. By Eq. (4), it holds[2]: $(pre^{-}_{F} \circ \gamma)(w^\sharp)(B) = \min_{a \in \mathcal{A}(B)} (pre[a]^{-}_{F} \circ \gamma)(w^\sharp)(B)$. Since the best transformer $f := \alpha^{\mathsf{u}} \circ (pre[a]^{-}_{F}) \circ \gamma$ is a valid abstraction of $pre[a]^{-}_{F}$, we have $(pre^{-}_{F} \circ \gamma)(w^\sharp)(B) \leq \min_{a \in \mathcal{A}(B)} (\gamma \circ f)(w^\sharp)(B)$. To finish the proof we exploit that γ is a morphism [23, Lemma 4.22] w.r.t. $\bigsqcup$:

$$\min_{a \in \mathcal{A}(B)} (\gamma \circ f)(w^\sharp)(B) = \gamma(\min_{a \in \mathcal{A}(B)} (f)(w^\sharp)(B)) = (\gamma \circ \widetilde{pre}^{\mathsf{u}-}_{F^\sharp})(w^\sharp)(B) \ . \qquad \blacksquare$$

4.3 Parallel-Abstraction Games

In this section we introduce *parallel-abstraction games*, the game construction corresponding to parallel abstraction.

Definition 6. *Let* $\mathcal{M} = (S, I, \mathcal{A}, R)$ *be an MDP and* Q *a partition. The parallel-abstraction game is* $\widehat{\mathcal{G}}_{\mathcal{M},Q} = ((V, E), V_{init}, (V_1, V_2, V_p), \delta)$ *with* $V_1 = Q \cup \{\star\}$, $V_2 = \{(v_1, a) \mid v_1 \in V_1, a \in \mathcal{A}(v_1)\}$, $V_p = \{\overline{\pi_{(s,a)}} \mid s \in S, a \in \mathcal{A}(s)\} \cup \{v^\star_p\}$, $V_{init} = \{B \in Q \mid B \cap I \neq \emptyset\}$, δ *is the identity function. Let* $v^\star_p(\star) = 1$. *The edges* E *are defined by:*

$$\begin{aligned} E = &\{(v_1, v_2) \mid v_1 \in V_1, v_2 = (v_1, a) \in V_2, a \in \mathcal{A}(v_1)\} \\ &\cup \{(v_2, v_p) \mid v_2 = (v_1, a) \in V_2, \exists s \in v_1 : \ v_p = \overline{\pi_{(s,a)}}\} \\ &\cup \{(v_2, v^\star_p), (v^\star_p, \star) \mid v_2 = (v_1, a) \in V_2, \exists s \in v_1 : \ a \notin \mathcal{A}(s)\} \\ &\cup \{(v_p, v') \mid v_p \in V_p, v_p(v') > 0\} \ . \end{aligned}$$

[2] Let $s_B \in S$ such that $B = \overline{s_B}$, obviously, $(pre^{-}_{F} \circ \gamma)(w^\sharp)(B) = (pre^{-}_{F})(\gamma(w^\sharp))(s_B)$. Applying Eq. (4), it can be rewritten to $\min_{a \in \mathcal{A}(B)} pre[a]^{-}_{F}(\gamma(w^\sharp))(B)$ which is the same as $\min_{a \in \mathcal{A}(B)} (pre[a]^{-}_{F} \circ \gamma)(w^\sharp)(B)$.

A player 1 vertex v_1 has a player 2 successor for each $a \in \mathcal{A}(v_1)$. A player 2 vertex (v_1, a) represents the abstraction of the a-transitions. Further, the partition may contain both states on which a particular action a is enabled and states on which it is not, i.e. the abstraction loses information about enabledness. In this case, player 2 vertex (v_1, a) has distribution $v_p^\star$ as a successor.

Example 2. We consider an MDP with states: $S = \{s_0, \ldots, s_7\}$. The state partition is $Q = \{\{s_0, s_1\}, \{s_2\}, \{s_3\}, \{s_4, s_7\}, \{s_5, s_6\}\}$. Figure 4 shows the MDP and the corresponding parallel-abstraction game.

The enabled actions $\mathcal{A}(B_0)$ in $B_0 = \{s_0, s_1\}$ are given by $\mathcal{A}(B_0) = \{a, b\}$. In the corresponding abstract game, blocks are player 1 vertices. For each enabled action, there is one player 2 vertex. For example, for block B_0, there are two player 2 vertices for action a and b respectively. The successors of the player 2 vertex for a reflect that from B_0 there are concrete a-transitions into $\{s_2\}$ and $\{s_3\}$. One successor of the player 2 vertex for b represents the b-distribution out of s_0. Vertex $\star$ reflects that b is not enabled at s_0. ■

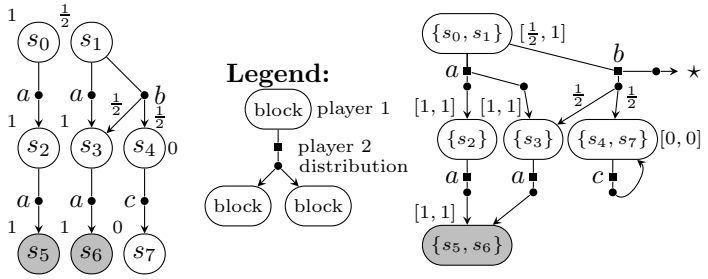


Fig. 4. Illustration of parallel abstraction.

Theorem 4 (Parallel-Abstraction Game and Parallel Abstraction). *Let $\mathcal{M}$, Q and $\widehat{\mathcal{G}}_{\mathcal{M},Q}$ as defined in Def. 6. Moreover, let $pre^{\pm\pm}_{V'}$ be the valuation transformers in $\mathcal{G}_{\mathcal{M},Q}$ w.r.t. objective V' as defined in Def. 2. Then we have: $\widetilde{pre}^{\mathfrak{l}-}_{F^\sharp} = pre^{--}_{F^\sharp \cup \{\star\}}$, $\widetilde{pre}^{\mathfrak{u}-}_{F^\sharp} = pre^{-+}_{F^\sharp \cup \{\star\}}$, $\widetilde{pre}^{\mathfrak{l}+}_{F^\sharp} = pre^{+-}_{F^\sharp}$ and $\widetilde{pre}^{\mathfrak{u}+}_{F^\sharp} = pre^{++}_{F^\sharp}$.*

While the proof proceeds in a similar fashion as Thm. 2, the following example provides some intuition.

Example 3. Figure 4 illustrates the concrete and abstract transformers for minimal reachability. In the MDP the minimal reachability probabilities w.r.t. goal states $\{s_5, s_6\}$ are written next to each state. In the game the probability bounds are annotated at each block as an interval, e.g., $[\frac{1}{2}, 1]$ for $\{s_0, s_1\}$. Now the function of the $\star$-vertex becomes clear. If there was no $\star$-vertex, action b would contribute probability $\frac{1}{2}$ and win against the probability 1 from a, so that for block $\{s_0, s_1\}$ the abstraction would spuriously report $\frac{1}{2}$ as an upper bound. ■

To compute the games for a given set of predicates, we employ SMT-based enumeration along the lines of [17, 2] with a few additions. The construction also has the same complexity (number of SMT solver calls). We refer to [2] for details,

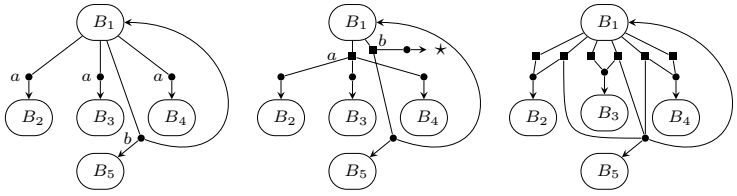
and focus on background needed to continue our exposition. Let $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ be a program, and let $\mathcal{M}$ be the semantics of $\mathtt{P}$. A predicate φ stands for the set of states satisfying it, i.e., $[\![\varphi]\!] \subseteq S$. A set of predicates $\mathcal{P}$ induces a finite partition of S: two states are in the same block iff they satisfy the same predicates.

Discussion and Comparison. Figure 5 shows a program. Consider predicates $s = 0 \dots 2, x < 0, x = 0$ and $x > 0$ and the following blocks in the partition: $B_1 = \{s = 0, x > 0\}$, $B_2 = \{s = 1, x < 0\}$, $B_3 = \{s = 1, x = 0\}$, $B_4 = \{s = 1, x = 0\}$ and $B_5 = \{s = 2, x > 0\}$.

```
module main
s    : [0..2];    // control flow
x,y : int;        // integer variables
[a] s=0 -> 1.0:(s'=1)&(x'=y);
[b] s=0 & x>10 -> 0.5:(s'=0)+ 0.5:(s'=2);
endmodule
```

Fig. 5. Example program.

Figure 6 shows MDP-based abstraction [2] (6(a)), parallel abstraction (6(b)) and game-based abstraction [16] (6(c)). The MDP-based abstraction contains four distributions. Command a induces three of them, since it assigns y to x, and there are states in B_1 where y is smaller, equal or less than zero respectively. Command b induces one distribution. Parallel abstraction has additionally two player 2 vertices, one for each command, so one can tell which command induces which distributions. Again, the $\star$-vertex reflects that command b is not enabled on all states in B_1. Game-based abstraction introduces six player 2 vertices: these vertices contain abstract distributions from *both* commands.



(a) MDP-based (b) Parallel (c) Game-based

Fig. 6. Example of parallel, MDP-based and game-based abstraction.

An advantage of parallel and MDP-based abstraction is that they can be computed by abstracting commands in isolation. Game-based abstraction, on the other hand, requires a different approach in presence of concurrency: certain player 2 vertices result from the combined effect of different commands. As a result, the abstraction needs to track correlations between different commands, which can be expensive [18]. Put differently, the number of player 2 vertices is the sum $\sum_{v_1 \in V_1} |\mathcal{A}(v_1)|$ in parallel abstraction, while, for game-based abstraction [16], this number is $\sum_{v_1 \in V_1} \left|\{\overline{Distr(s)} \mid s \in v_1\}\right|$ and, since each player 2 vertex corresponds to a *subset* of V_p, the worst case lies in the order of $2^{|V_p|}$.

4.4 Refinement

We are interested in (minimal or maximal) reachability probabilities for the initial states of the program. The analysis produces lower and upper bounds w^{l} and w^{u} respectively for the reachability probabilities w.r.t. to goal states F. Refinement is invoked whenever the bounds for the initial vertices are too

imprecise, i.e. $w^{\mathrm{u}}(v_{init}) - w^{\mathrm{l}}(v_{init}) > \varepsilon$ for some initial game vertex $v_{init} \in V_{init}$ where ε is the user-specified precision.

The analysis of the game also yields a game strategy for the lower and upper bound respectively. We can use existing techniques [2] to check if abstract paths, ending with a goal block and admissible w.r.t. the strategy, actually correspond to feasible paths in the program. However, in the given framework, we can focus feasibility checks on parts of the game where the bounds are not tight.

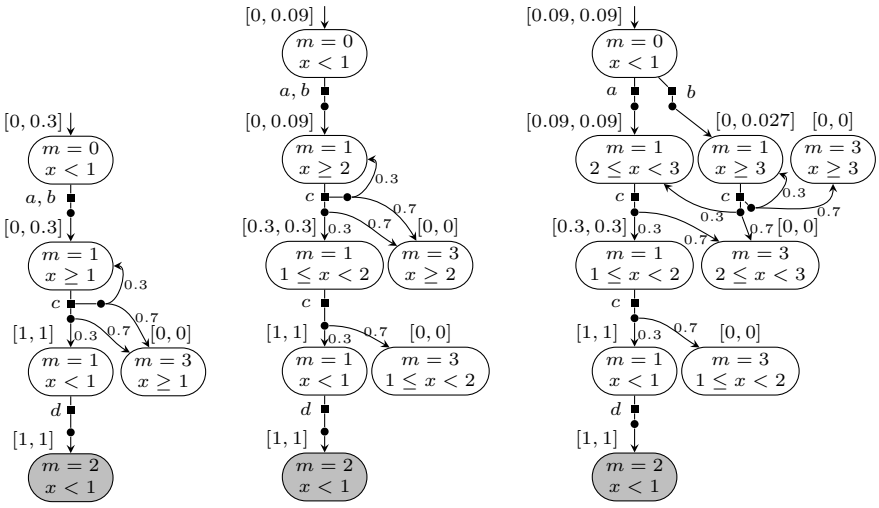
The idea is to refine blocks $v_1 \in V_1$ containing states $s \in v_1$ that would achieve more precise bounds if separated from the other states in the block, i.e., more formally, $\left((pre_F \circ \gamma)(w^{\mathrm{l}})\right)(s) < w^{\mathrm{u}}(v_1)$ or $w^{\mathrm{l}}(v_1) < \left((pre_F \circ \gamma)(w^{\mathrm{l}})\right)(s)$ (where pre_F is the concrete transformer, i.e. pre_F^- for minimal and pre_F^+ for maximal reachability). For example, consider the program in Figure 3 and its abstraction using predicates $\{m = 0, m = 1, m = 2, m = 3, x \geq 1\}$ in Figure 7(a). We want to compute the maximal probability to reach $\{m = 2\}$. Consider block $B = \{m = 1, x \geq 1\}$. All states with $x < 1$ in B (in this case just one state s with $m = 1, x = 1$) can go to the block $\{m = 1, x < 1\}$ via c with probability 0.3. Thus state s fulfills $\left((pre_F \circ \gamma)(w^{\mathrm{l}})\right)(s) = \left((pre_F \circ \gamma)(w^{\mathrm{u}})\right)(s) = 0.3$. By introducing the predicate $\{x \geq 2\}$, we obtain the abstraction in Figure 7(b) where block B is split into blocks $\{m = 1, x \geq 2\}$ and $\{m = 1, 1 \leq x < 2\}$ with more precise probability bounds $[0.3, 0.3]$ and $[0, 0.09]$ respectively.

We put these ideas to work in the following way. First, we select a block to be refined such that[3]: (1) lower and upper bound differ $w^{\mathrm{u}}(v_1) - w^{\mathrm{l}}(v_1) > 0$ and, (2) for a player 2 vertex $v_2 \in E(v_1)$, the lower-bound strategy chooses some $v \in E(v_2)$ distinct from the choice of the upper-bound strategy $v' \in E(v_2)$. Then we invoke function RefBlock$(v_1, (v, v'))$. If v or v' equals v_p^*, RefBlock returns the guard of command a. Otherwise, there exists $v_1' \in V_1$ such that $v(v_1') \neq v'(v_1')$ where v and v' differ in a predicate valuation for some predicate φ. Then RefBlock returns the precondition of φ w.r.t. command a. In the example, we have computed preconditions of predicates $\{x \geq 1\}$ and $\{x \geq 2\}$ w.r.t. command c and update `x'=x-1` which leads to the refinement steps in Figure 7(b) and 7(c). This refinement strategy is inspired by [3]: while they consider sequential programs we consider concurrent programs.

5 Experiments

We have implemented our method in the `PASS` tool [2], which, until recently, gave only upper bounds for *maximal* probabilistic reachability, and, only in some cases, effective lower bounds from counterexample analysis. The new version `PASS 2.0` provides both lower and upper bounds for minimal and maximal probabilistic reachability. Experiments were run on a Pentium 4 with 2.6GHz and 1GB RAM. We considered models of network protocols, including all models from [2] and, examples of probabilistic C programs from [3], if they could be translated to `PASS` models. We first discuss minimal reachability problems (here `PASS 1.0` is not applicable): `PASS 2.0` computed precise *minimal* reachability probabilities for properties of the `csma` and `wlan` models from [2]. Further, it

[3] This is a relaxation of the criterion that refers to the concrete transformer.



(a) Step 1. (b) Step 2: $x \geq 2$. (c) Step 3: $x \geq 3$.

Fig. 7. Illustration of abstraction refinement.

solved the `zeroconf` and `herman` case study from [3]. Their tool took 1.97s and 33.5s respectively, on a faster machine, compared to 1.3s and 5s for `PASS 2.0`. In the table below, we compare with `PASS 1.0` giving running times (in seconds):

	`wlan1`		`wlan2`		`csma1`		`csma2`		`brp1`		`brp2`		`sw1`		`sw2`	
`PASS 2.0`	43s	✔	115s	✔	10s	✔	5s	✔	27s	✔	1s	✔	18s	✔	2s	✔
`PASS 1.0`	72s	✗/✔	306s	✗/✔	38s	✗/ ✔	11s	✗/ ✔	21s	✔	3s	✔	87s	90%/45%	89s	✔

In the table ✔ means success, i.e. the difference between the two established bounds is less than $\varepsilon = 10^{-6}$. ✗/ ✔ means lower bound 0 and a correct upper bound. 90%/45% means 90% underestimation of the lower and 45% overestimation of the upper bound. `PASS 2.0` succeeds in all cases, while `PASS 1.0` successfully finds upper bounds, in one case, however, an imprecise one. `PASS 2.0` is often faster. Thanks to lower and upper bounds, it focuses on points of imprecision and thus finds smaller abstractions.

6 Conclusion

Abstraction is the key to probabilistic model checking of realistic models. This paper presents the first abstract-interpretation framework for MDPs which admits to compute both lower and upper bounds on reachability probabilities. Based on this framework, we present an automatic analysis for concurrent programs to compute precise lower and upper bounds on reachability probabilities. As future work, we would like to extend our framework also to probabilistic safety, rewards and probabilistic equivalence checking [24, 25].

Acknowledgements. This work was supported by the DFG as part of SFB/TR 14 AVACS, by the European Community's FP7 under grant n^o 214755, and the NWO-DFG bilateral project ROCKS. We would like to thank the anonymous reviewers, Claire Burguière and Ernst Moritz Hahn for their comments.

References

1. Bianco, A., de Alfaro, L.: Model Checking of Probabilistic and Nondeterministic Systems. In: FSTTCS, Springer (1995) 499–513
2. Hermanns, H., Wachter, B., Zhang, L.: Probabilistic CEGAR. In: CAV. (2008) 162–175
3. Kattenbelt, M., Kwiatkowska, M.Z., Norman, G., Parker, D.: Abstraction Refinement for Probabilistic Software. In: VMCAI. (2009) 182–197
4. Cousot, P., Cousot, R.: Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints. In: POPL. (1977) 238–252
5. Ball, T., Podelski, A., Rajamani, S.K.: Boolean and Cartesian Abstraction for Model Checking C Programs. In: TACAS. (2001) 268–283
6. Cousot, P., Cousot, R., Feret, J., Mauborgne, L., Miné, A., Monniaux, D., Rival, X.: The ASTREÉ Analyzer. In: ESOP. (2005) 21–30
7. Alt, M., Ferdinand, C., Martin, F., Wilhelm, R.: Cache Behavior Prediction by Abstract Interpretation. In: SAS. (1996) 52–66
8. Reps, T.W., Sagiv, S., Yorsh, G.: Symbolic Implementation of the Best Transformer. In: VMCAI. (2004) 252–266
9. Cousot, P., Cousot, R.: Systematic Design of Program Transformation Frameworks by Abstract Interpretation. In: POPL '02, ACM (2002) 178–190
10. Monniaux, D.: Abstract Interpretation of Programs as Markov Decision Processes. Sci. Comput. Program. **58** (2005) 179–205
11. Pierro, A.D., Wiklicky, H.: Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation. In: PPDP. (2000) 127–138
12. Smith, M.J.A.: Probabilistic Abstract Interpretation of Imperative Programs using Truncated Normal Distributions. ENTCS **220** (2008) 43–59
13. Coletta, A., Gori, R., Levi, F.: Approximating Probabilistic Behaviors of Biological Systems Using Abstract Interpretation. ENTCS **229** (2009) 165–182
14. Monniaux, D.: Backwards Abstract Interpretation of Probabilistic Programs. In: ESOP. (2001) 367–382
15. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: TACAS. (2006) 441–444
16. Kwiatkowska, M.Z., Norman, G., Parker, D.: Game-based Abstraction for Markov Decision Processes. In: QEST. (2006) 157–166
17. Wachter, B., Zhang, L., Hermanns, H.: Probabilistic model checking modulo theories. In: QEST. (2007) 129–140
18. Kattenbelt, M., Kwiatkowska, M.Z., Norman, G., Parker, D.: Game-Based Probabilistic Predicate Abstraction in PRISM. In: QAPL. (2008)
19. D'Argenio, P.R., Jeannet, B., Jensen, H.E., Larsen, K.G.: Reachability Analysis of Probabilistic Systems by Successive Refinements. In: PAPM-PROBMIV. (2001) 39–56
20. Tarski, A.: A Lattice-Theoretical Fixpoint Theorem and Its Applications. In: Pacific Journal of Mathematics 5:2:. (1955) 285–309
21. Baier, C.: On Algorithmic Verification Methods for Probabilistic Systems (1998) Habilitationsschrift, Universität Mannheim.
22. Condon, A.: The Complexity of Stochastic Games. Inf. Comput. **96** (1992) 203–224
23. Nielson, F., Nielson, H.R., Hankin, C.: Principles of Program Analysis. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1999)
24. Murawski, A.S., Ouaknine, J.: On Probabilistic Program Equivalence and Refinement. In: CONCUR. (2005) 156–170
25. Legay, A., Murawski, A.S., Ouaknine, J., Worrell, J.: On Automated Verification of Probabilistic Programs. In: TACAS. (2008) 173–187