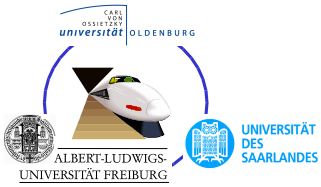


AVACS – Automatic Verification and Analysis of Complex Systems

REPORTS

of SFB/TR 14 AVACS

Editors: Board of SFB/TR 14 AVACS

Best Probabilistic Transformers and Beyond

by

Björn Wachter, Lijun Zhang

AVACS Technical Report No. 48
April 2009
ISSN: 1860-9821

Publisher: Sonderforschungsbereich/Transregio 14 AVACS
(Automatic Verification and Analysis of Complex Systems)
Editors: Bernd Becker, Werner Damm, Martin Fränzle, Ernst-Rüdiger Olderog,
Andreas Podelski, Reinhard Wilhelm
ATRs (AVACS Technical Reports) are freely downloadable from `www.avacs.org`

Author(s) contact: Björn Wachter (`bwachter@cs.uni-sb.de`).

Best Probabilistic Transformers and Beyond

Björn Wachter, Lijun Zhang

Saarland University, Saarbrücken, Germany
{bwachter,zhang}@cs.uni-sb.de

Abstract. How can analyses for probabilistic systems be achieved that are optimal, in the sense that they enjoy the best possible precision for a given abstraction? Aiming at the problem at the heart of many algorithms for probabilistic model checking – computing the probability of reaching a particular set of states – we leverage the theory of abstract interpretation, obtaining, for the first time, a characterization of such optimal analyses in this setting. With a focus on concurrent system and predicate abstraction, the paper gives ways to efficiently compute, approximate and refine such abstractions, and an experimental evaluation.

1 Introduction

Probabilistic models are used to analyze systems with random phenomena like network protocols, and randomized algorithms. The underlying semantics are Markov Decisions Processes (MDPs), an extension of transition systems featuring non-deterministic and probabilistic choice. Typically one is interested in computing *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 abstraction refinement techniques have evolved [2, 3] that scale to larger models, like programs. They build compact abstract programs via predicate abstraction. While for the non-probabilistic case, such constructions are well-studied [4], and have a foundation [5] in the theory of abstract interpretation, this is not so for the probabilistic case. And indeed questions remain open, e.g. for given predicates, what is the most precise abstract program that is still a valid abstraction?

The fixpoint characterization of probabilistic reachability [6], not only forms the basis of algorithms in current model checkers [7, 2, 3], it also opens the way to apply the Cousots' theory of abstract interpretation (AI) [8], where the program semantics, described as the fixpoint of a functional, is approximated by an abstract fixpoint. Thereby, the program is given an interpretation over the abstract domain that safely approximates the original semantics. Further, a specification of the optimal analysis is given by composition of concretization function γ, the functional f characterizing the program semantics and abstraction function α:

$$f^\sharp = \alpha \circ f \circ \gamma$$

under the condition that functions α and γ form a Galois connection [9]. Functional $f^\sharp$, called *best transformer*, is the limit on the best achievable precision. This concept is the starting point for our work.

Best transformers are *the* reference point for the design of abstraction-based analyses [10, 5, 11, 12]. However they are uncharted territory in the area of MDPs, although several approaches [13–16] apply AI to probabilistic systems. Closely related is work by Monniaux [13] on computing upper bounds on probabilistic reachability for infinite MDPs. Where the abstract domain is the space of linear combinations over a base domain. The framework allows to plug in a whole wealth of base domains from static analysis. However, by construction, the abstract domain does not support Galois connection and best transformers because it is not partially ordered [17], due to how the lattice structure of base domains interacts with linear combinations. We introduce a new theory that supports Galois connections and best transformers. Our new theory is not an instance of Monniaux's [13, 17] – our concrete and abstract domains are vector spaces with a Galois connection between them – or of Probabilistic AI [14], which also considers vector spaces, however targets models without non-determinism, like other more recent work [15, 16]. Neither is it a replacement of [13, 17], since its base domain are state partitions, which fits with predicate abstraction, but not with classical base domains from static analysis.

While the AI-based approaches of [13–16] have yet to be combined with automatic abstraction refinement, predicate abstraction has recently shown promising results. The abstraction-refinement method Probabilistic CEGAR [2] computes upper bounds for *concurrent probabilistic programs*, an infinite-state variation of the language of the popular probabilistic model checker `PRISM` [7]. The software model-checking tool presented in [3] employs predicate abstraction and, going beyond single-sided bounds, use an abstraction based on games [18], *game-based abstraction*, that yields both lower and upper bounds.

Our first contribution, is the first formalization of abstract probabilistic reachability analysis for MDPs with Galois connections and best transformers. Revealing the connection between the denotational concept of best transformers, which is rather declarative, and the operational concept of game-based abstraction [18], which is constructive, we are able to show how to compute best transformers and that there are no transformers more precise than game-based abstraction. Yet, computing best transformers can incur prohibitive cost: while, in sequential programs, commands can be abstracted separately, as in [3], in concurrent programs, parallel composition requires expensive tracking of correlations between commands, which calls for approximations. Our second contribution is an abstraction refinement technique for concurrent probabilistic programs that computes best transformers per command and refines lost correlations lazily as needed. It is complete, i.e., by adding predicates, the precision of the best transformer can be achieved (unlike the orthogonal approach of [19]). Its implementation produces precise lower and upper bounds with competitive speed.

Outline. Background in AI and MDPs is given in Section 2. We continue with abstractions for probabilistic reachability in Section 3 culminating in the result that game-based abstraction computes best transformers. Abstraction refinement of concurrent probabilistic programs using best transformers is described in Section 4. Experimental results are given in Section 5.

2 Background

Section 2.1 first recalls some AI basics like lattices and best transformers. We then introduce the lattice of valuations, the domain for abstract probabilistic reachability, which admits expressing lower and upper bounds. In Section 2.2, we turn to MDPs and the fixpoint characterization of probabilistic reachability.

2.1 Galois connections, Best Transformers and Valuations

The pair $(A, \leq)$ is a partially-order 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 say $f \leq g$ 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 has a unique least element and a unique greatest element with respect to $\leq$, i.e., there exist functions $\bigsqcap, \bigsqcup : 2^L \to L$ such that $\bigsqcap L' \leq l \leq \bigsqcup L'$ for all $l \in L' \subseteq L$.

For a monotone function $f : L \to L$ over a lattice $(L, \leq)$, Tarski's theorem [20] guarantees existence of least and greatest fixpoints, *lfp* f and *gfp* f respectively. Further, f is called a $\bigsqcap$-morphism ($\bigsqcup$-morphism) if, for all $M \subseteq L$, $f(\bigsqcap M) = \bigsqcap\{f(l) \mid l \in M\}$ ($f(\bigsqcup M) = \bigsqcup\{f(l) \mid l \in M\}$ respectively).

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 effect of a command 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* [8], 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, i.e. real numbers between zero and one. The valuations over S form a lattice $(W_S, \leq)$ where $W_S = \{w \mid w : S \to [0, 1]\}$ and $\leq = \{(w, w') \mid \forall s \in S : \ w(s) \leq w'(s)\}$. Figure 1 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 $(W_S, \geq)$ is obtained by inverting the 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 $V \subseteq W_S$.

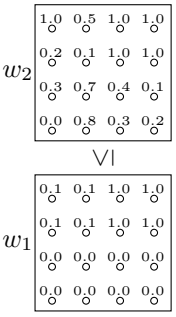


Fig. 1.

A *valuation transformer* is a monotone function $f : W_S \to W_S$. We refer to the least fixpoint of f over $(W_S, \leq)$ by $lfp_{\leq}\ f$, to the greatest fixpoint over $(W_S, \geq)$ by $gfp_{\geq}\ f$ and analogously for the other cases. Due to duality, we have $(gfp_{\geq}\ f) = (lfp_{\leq}\ f)$ and $(lfp_{\geq}\ f) = (gfp_{\leq}\ f)$.

2.2 Markov decision processes

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

A *Markov decision process* (MDP) $\mathcal{M}$ is a tuple (Σ, I, A, R) where Σ is a set of states, $I \subseteq \Sigma$ is a set of initial states, A is a finite action alphabet, and $R : \Sigma \times A \rightharpoonup Distr_\Sigma$ the transition function. We say $\mathcal{M}$ is finite if Σ is finite.

In a state of the MDP, possibly only a subset of the actions is enabled, therefore R is a partial function, as indicated by the $\rightharpoonup$ arrow. For a state $s \in \Sigma$, denote the enabled actions by $A(s) = \{a \mid \exists \pi \in Distr_\Sigma.\ \pi = R(s, a)\}$, and out-going distributions by $Distr(s) = \{R(s, a) \mid a \in A(s)\}$. We define a total function $\pi_{\cdot,\cdot} : \Sigma \times A \to Distr_\Sigma$, where $\pi_{(s,a)} = R(s, a)$ if $a \in A(s)$, and otherwise $\pi_{(s,a)} = 0_\Sigma$. We say that $(s, a, \pi_{(s,a)})$ is a transition of $\mathcal{M}$ if $a \in A(s)$.

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 $i \in \mathbb{N}_0$. Let $Path(\mathcal{M})$ denote the set of all paths over $\mathcal{M}$. Similarly finite paths can be defined. For $\beta \in Path$, let $\beta[i] = s_i$ denote the $i + 1$-th state of β.

For an MDP $\mathcal{M}$ and a subset of its action $A' \subseteq A$, we define the sub-MDP $\mathcal{M}_{A'} = (\Sigma, I, A', R_{A'})$ induced by A'. Thereby the transition function is restricted to actions from A' such that $R_{A'} : \Sigma \times A' \rightharpoonup Distr_\Sigma, (s, a) \mapsto R(s, a)$.

Markov chains (MCs) are special cases of MDPs, deterministic MDPs where for every state s there is at most one enabled transition $|A(s)| \leq 1$. Unlike a MC, 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 non-determinism.

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 $\Sigma_{\mathcal{M}}$. A strategy σ is called *simple* if it does not use randomization and is memoryless, i.e. if it is a function $\sigma : \Sigma \to A$. For a given state $s \in \Sigma$ and a strategy σ, let P_s^σ denote the corresponding probability measure [21] over $Path(\mathcal{M})$. Given an MDP $\mathcal{M}$, a simple strategy σ induces a MC as follows: $\mathcal{M}_\sigma = (\Sigma, I, A, R_\sigma)$ where $R_\sigma = \{(s, a, \pi) \in R \mid \pi = R(s, \sigma(s))\}$.

Probabilistic Reachability. Let $\mathcal{M} = (\Sigma, I, A, R)$ be an MDP, and let $F \subseteq \Sigma$ be a set of goal states. The probability measure P_s^σ induced by strategy σ gives, for each state s, the probability $p_s^\sigma(F)$ of reaching a goal state:

$$p_s^\sigma(F) = P_s^\sigma(\{\beta \in Path(\mathcal{M}) \mid \exists i \in \mathbb{N}_0 : \ \beta[i] \in F\}) \quad \text{for each } s \in \Sigma \ .$$

For a fixed strategy σ, this defines a valuation $p^\sigma(F) \in W_\Sigma$ which maps a state s to $p_s^\sigma(F)$. In the context of MDPs, one studies minimal $p^-(F) \in W_\Sigma$ and maximal $p^+(F) \in W_\Sigma$ 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. There always exist *simple* strategies that are optimal [1].

Baier [6] showed that minimal and maximal reachability probabilities are expressible as *least fixpoints* of valuation transformers. Let $F_0 \subseteq \Sigma$ be states that cannot reach states in F. Minimal reachability $p^-(F)$ is the *least fixpoint* $lfp_\leq\ pre_F^-$ of the valuation transformer $pre_F^- : W_\Sigma \to W_\Sigma$ where pre_F^- is defined by $pre_F^-(w)(s) = 1$ if $s \in F$, $pre_F^-(w)(s) = 0$ if $s \in F_0$, and for all other states: $pre_F^-(w)(s) = \min_{a \in A(s)} \sum_{s' \in \Sigma} \pi_{(s,a)}(s') \cdot w(s')$. The valuation transformer $pre_F^+ : W_\Sigma \to W_\Sigma$ is defined analogously, with the difference that it maximizes over all enabled actions. We have $p^+(F) = lfp_\leq\ pre_F^+$.

This fixpoint characterization of probabilistic reachability together with the background on AI allows us to express abstractions for probabilistic reachability.

3 Abstraction

We present our framework for MDP abstraction and probabilistic reachability following the best-transformer paradigm of Cousot [9]. Several novel aspects come into play which set this apart from previous work by Monniaux [13]: (1) the abstract domains we consider are lattices which allows us to develop best transformers in the first place (2) we consider lower and upper bounds on probabilistic reachability, and (3) our concrete transformers are exactly the valuation transformers corresponding to minimal and maximal reachability, not lifted transformers over power set of valuations. In Section 3.1, we develop the abstract domain and define best transformers. These turn out to be reminiscent of the alternation in two-player games, which we recall in Section 3.2. In Section 3.3, we come to one of the central results of the paper: we show how to compute best transformers by game-based abstraction.

3.1 Lower- and Upper-Bound Abstraction

Let Σ be a set of states, e.g. of an MDP. A partition Q of Σ is a set of pairwise disjoint, nonempty subsets of Σ such that $\bigcup_{B\in Q} B = \Sigma$. Elements of Q are called blocks. For a state $s \in \Sigma$, we denote by $\overline{s}$ the unique block B containing state 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* $\alpha^- : W_\Sigma \to W_Q$ returns the infimum of the values $\alpha^-(w)(B) = \inf_{s\in B} w(s)$ within a block while *upper-bound abstraction* $\alpha^+ : W_\Sigma \to W_Q$ returns the supremum $\alpha^+(w)(B) = \sup_{s\in B} w(s)$.

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 $\gamma : W_Q \to W_\Sigma$. Although there is just one concretization, the interpretation of a valuation depends on the lattice order: where $\geq$ is the order for lower bounds, and, $\leq$ the one for upper bounds. The inversion reflects the duality that a lower bound is the more precise the larger it is, while the converse is true for an upper bound.

We obtain two Galois connections corresponding to α^- and α^+ respectively:

Proposition 1 (Galois Connections). *For a given partition Q, where α^-, α^+, γ are the functions defined above, we have the Galois connections:*

$$(W_\Sigma, \geq) \underset{\alpha^-}{\overset{\gamma}{\leftrightarrows}} (W_Q, \geq) \qquad (W_\Sigma, \leq) \underset{\alpha^+}{\overset{\gamma}{\leftrightarrows}} (W_Q, \leq)$$

Discussion. The two Galois connections are motivated by the different interpretation of valuations. As illustrated in Figure 2, for a valuation w over states, here marked by a thick border, lower-bound abstraction $\alpha^-(w)$ provides a lower bound, i.e. valuation $\gamma(\alpha^-(w))$ over states, is below w; the value for each state s is less equal $w(s)$. However, if we do the same taking the α^+-abstraction, the resulting valuation over states $\gamma(\alpha^+(w))$ is an upper bound for valuation w. The big dashed box on the left represents valuations over states where w is the valuation with the thick border. The partition into blocks B_1, B_2, B_3 is depicted by rectangles surrounding states. The dashed box on the right represents valuations over blocks.

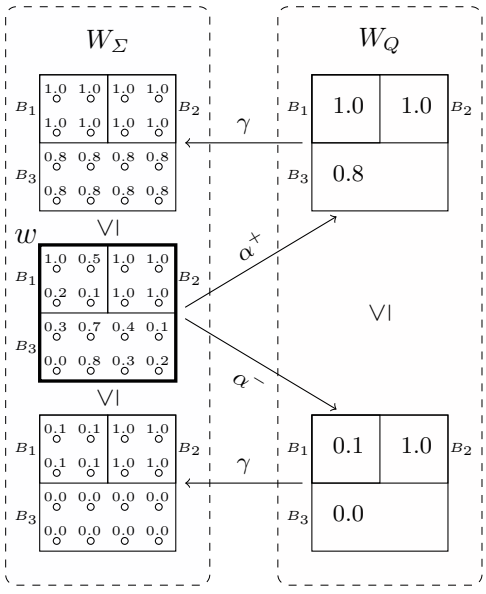


Fig. 2. Illustration of α^- and α^+.

Bounds for Probabilistic Reachability. We obtain four different best transformers $\alpha^{\pm} \circ pre_F^{\pm} \circ \gamma$ where the abstraction function controls whether we get lower or upper bounds, and the valuation transformer controls the interpretation of non-determinism, i.e., whether the maximal or minimal reachability probability are considered. The connection to probabilistic reachability is established by:

Lemma 1. *Fixpoints of the best transformers yield lower and upper bounds on minimal and maximal reachability probabilities:*

$$\gamma(gfp_{\geq}(\alpha^{-} \circ pre_F^{\oplus} \circ \gamma)) \leq p^{\oplus}(F) \leq \gamma(lfp_{\leq}(\alpha^{+} \circ pre_F^{\oplus} \circ \gamma)) \quad \textit{where } \oplus \in \{-,+\} \ .$$

The definition of the best transformer for the upper of minimal reachability $(\alpha^{+} \circ pre_F^{-} \circ \gamma)$ contains an alternation between minimization, as in pre_F^{-}, and maximization, as in α^{+}. This suggests a connection to games where minimization and maximization are the objectives of two adversarial players. After a brief interlude, recalling probabilistic games in Section 3.2, the connection to game-based abstraction is made in Section 3.3 which provides a scheme for computing best transformers.

3.2 Turn-based probabilistic games

A *turn-based probabilistic game* [22] is a tuple $\mathcal{G} = ((V,E), V_{init}, (V_1, V_2, V_p), \delta)$ where (V,E) is a 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 $i \in \mathbb{N}_0$. Let $Play(\mathcal{G})$ denote the set of all plays. Let $\omega[i] = s_i$ denote the $i+1$-th vertex of ω, and $last(\omega) = v_n$ denote the last vertex.

A player 1 strategy is a function $\sigma_1 : V^* V_1 \to Distr_V$ such that for any play $\omega \in Play(\mathcal{G})$, $\sigma_1(\omega)(v) > 0$ 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 σ_1, σ_2 induces a MC, denoted by $\mathcal{G}_v^{\sigma_1,\sigma_2}$, and a corresponding probability measure $P_v^{\sigma_1,\sigma_2}$ [22].

Given a vertex v, a *reachability objective* $F \subseteq V$, and strategies σ_1, σ_2, $p_v^{\sigma_1,\sigma_2}(F)$ denotes the probability of reaching F starting from v:

$$p_v^{\sigma_1,\sigma_2}(F) = P_v^{\sigma_1,\sigma_2}(\{\omega \in Play(\mathcal{G}) \mid \exists i \in \mathbb{N}_0 : \ \omega[i] \in F\}) \ .$$

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: $\sup_{\sigma_1} \inf_{\sigma_2} p^{\sigma_1,\sigma_2}, \inf_{\sigma_1} \sup_{\sigma_2} p^{\sigma_1,\sigma_2} \in W_V$ where 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. Like in [18], we consider the cases where both players cooperate $\inf_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F)$ and $\sup_{\sigma_1,\sigma_2} p^{\sigma_1,\sigma_2}(F)$.

Valuation Transformers. Similar to MDPs, pure memoryless strategies already achieve the optima [23] and optimal valuations can be computed as least fixpoints of valuation transformers. For instance, $\sup_{\sigma_1} \inf_{\sigma_2} p^{\sigma_1,\sigma_2}(F) = lfp_{\leq}\ pre_F^{+-}$ with $pre_F^{+-} : W_{V_1} \to W_{V_1}$ and for all valuations d: $pre_F^{+-}(d)(v)$ equals 1 if $v \in F$ and 0 if $v \in F_0$, and otherwise by $\max_{v_2 \in E(v)}\ \min_{v_p \in E(v_2)} \sum_{v' \in E(v_p)} d(v') \cdot \delta(v_p)(v')$. 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 $E(v_2)$. These valuation transformers are used to effectively compute optimal valuations by value iteration [3, 18].

3.3 Best Transformers from Game-Based Abstraction

Intuitively, the roles of the players in game-based abstraction [18] are as follows: a V_1 vertex corresponds to a block. Player 1 picks a state from the block and then player 2 chooses an out-going distribution of the state. Since we operate over blocks the abstraction of this distribution is taken. The abstraction of a distribution $\pi \in Distr_\Sigma$ is the distribution $\overline{\pi} \in Distr_Q$ with $\overline{\pi}(B) := \sum_{s \in B} \pi(s)$. Lifting this to sets of distributions we get $\overline{D} = \{\overline{\pi} \mid \pi \in D\}$ for $D \subseteq Distr_\Sigma$.

Definition 1 ([18]). *Let $\mathcal{M} = (\Sigma, I, A, R)$ be an MDP, and let Q be a partition. The game abstraction of $\mathcal{M}$ induced by Q is the turn-based probabilistic game $\mathcal{G}_{\mathcal{M},Q} = ((V, E), V_{init}, (V_1, V_2, V_p), \delta)$ defined as follows: the player 1 vertices are given by the blocks $V_1 = Q$, the player 2 vertices $V_2 \subseteq 2^{Distr_Q}$ are defined as $V_2 = \{\overline{Distr(s)} \mid s \in \Sigma\}$. Vertices $V_p \subseteq Distr_Q$ are distributions $V_p = \{\overline{\pi} \mid \exists s \in S :\ \pi \in Distr(s)\}$. $\delta : V_p \to Distr_V$ 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:*

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

Re-stating the results of [18], the salient point is the following relation between the optimal valuations in the game and optimal reachability probabilities in $\mathcal{M}$. We assume that the partition Q is chosen such that the goal states F in $\mathcal{M}$ are exactly representable: i.e., $F = \bigcup_{B \in F^\sharp} B$ for a suitable $F^\sharp \subseteq Q$. Then we have:

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

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

In [18] this result was proved by showing that each strategy present in the MDP can be mapped to a game strategy with the same value.

We show (Theorem 1) that the best transformers with respect to lower and upper bound abstraction (defined in Section 3.1) are equal to the valuation transformers of the game. Recall, for a game and reachability objective, the optimal valuations are the fixpoints of the valuation transformers. Thus, using Lemma 1, we obtain another proof for Equations 1 and 2.

Theorem 1. *Let $\mathcal{M}$ be an MDP and $F \subseteq \Sigma$ a set of goal states. Further, consider the parition Q of Σ. Then the valuation transformers $pre^{\pm\pm}_{F^\sharp}$ induced by the game-based abstraction $\mathcal{G}_{\mathcal{M},Q}$ are the best transformers $\alpha^\pm \circ pre^\pm_F \circ \gamma$:*

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

where α^- and α^+ are the lower- and upper-bound abstractions, and γ the concretization function with respect to partition Q as defined in Subsection 3.1.

One can thus compute best transformers by game-based abstraction. Yet, the computational cost can be high. In predicate abstraction, abstract models are constructed by enumerating the abstract transformer with a decision procedure. V_2 vertices are *sets* of abstract distributions, leading to a number of enumerations exponential in V_p, i.e., $|V_2| = 2^{|V_p|}$, in the worst case. For the abstraction used in [2], enumerating abstract distributions, not sets of them, is enough. We would like to avoid this increase in complexity and still obtain lower and upper bounds. A solution for concurrent probabilistic programs is presented in Section 4.

4 Abstraction Refinement for Concurrent Programs

We present an abstraction refinement method that computes best transformers for each command of a concurrent program. After reviewing concurrent probabilistic programs, we present the analysis in Subsection 4.2 discussing the practical implications of the compositional approach and an efficient implementation based on predicate abstraction. We then give the refinement algorithm in Subsection 4.3. Experimental results are presented in Section 5.

4.1 Concurrent Probabilistic Programs

We consider concurrent probabilistic programs as in [2], a variation of the popular `PRISM` language [7] that additionally supports integer and real variables. We now introduce the abstract syntax and the semantics of this language. We fix a finite set of program variables $\mathtt{X}$ and a finite set of actions 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 $\mathtt{X}$. 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 "+": $[a]$ $\mathtt{g} \to \mathtt{p}_1 : \mathtt{X}'{=}E_{\mathtt{u}_1}$ + ... + $\mathtt{p}_k : \mathtt{X}'{=}E_{\mathtt{u}_k}$. If the guard is satisfied, the i-th update executes with probability p_i. For $\mathtt{c}$, write $a_{\mathtt{c}}$ for its action, $\mathtt{g}_{\mathtt{c}}$ for its guard. We omit subscripts if the command is clear from context. The weakest precondition of an expression with respect to an update $\mathtt{u}$ is: $\mathtt{WP}_{\mathtt{u}}(\mathtt{e}) = \mathtt{e}[\mathtt{X}/E_{\mathtt{u}}(\mathtt{X})]$ where all occurrences of variables are replaced according to $\mathtt{u}$.

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 $\Sigma(\mathtt{X})$, or Σ for short, and a single state by s. For an expression $\mathtt{e} \in \mathit{Expr}_{\mathtt{X}}$, we denote by $[\![\mathtt{e}]\!]_s$ its valuation in state s. For a Boolean expression $\mathtt{e} \in \mathit{BExpr}_{\mathtt{X}}$, we have $[\![\mathtt{e}]\!]_s \in \{0,1\}$ and denote by $[\![\mathtt{e}]\!] = \{s \in \Sigma \mid [\![\mathtt{e}]\!]_s = 1\}$ the states fulfilling $\mathtt{e}$.

The semantics of a program $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ is the MDP $\mathcal{M} = (\Sigma, I, A, R)$ with states $\Sigma = \Sigma(\mathtt{X})$, initial states $I = [\![\mathtt{I}]\!]$, actions $A = \{a_{\mathtt{c}} \mid \mathtt{c} \in \mathtt{C}\}$, and transitions induced by the commands. Consider $s \in \Sigma$ and $a \in A$. Let $\mathtt{c}$ be the corresponding command with $a = a_{\mathtt{c}}$. If $s \in [\![\mathtt{g}]\!]$, we define $R(s,a) = \pi$ 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 Compositional Abstraction

Let $\mathtt{P}$ be a program with semantics $\mathcal{M} = (\Sigma, I, A, R)$. We assume the partition Q is chosen such that there exists $F^\sharp \subseteq Q$ with $F = \bigcup_{B \in F^\sharp} B$. Furthermore, we define $A(B) := \{a \mid \exists s \in B : a \in A(s)\}$ for a block $B \in Q$.

In compositional abstraction, we first compute the best transformers of the commands separately and then combines these to an abstraction of the program. Commands behave deterministically and each command corresponds to a unique action a. For command a, the concrete transformer is $\mathit{pre}[a]_F \in W_\Sigma \to W_\Sigma$, regardless of minimal or maximal reachability due to the determinism of commands. Where $\mathit{pre}[a]_F$ refers to the valuation transformer in the sub-MDP induced by action set $\{a\}$ with respect to the goal states F. Further, best transformers $\mathit{pre}^\sharp[a]^{\oplus}_{F^\sharp} := \alpha^{\oplus} \circ (\mathit{pre}[a]_F) \circ \gamma$ are given for lower ($\oplus = -$) and upper ($\oplus = +$) bounds respectively.

We define valuation transformers $\mathit{pre}^{\pm\pm}_{F^\sharp} \in W_Q \to W_Q$. For all $w^\sharp \in W_Q$ and all blocks $B \in Q$, we have:

$$\mathit{pre}^{-\oplus}_{F^\sharp}(w^\sharp)(B) = \min_{a \in A(B)} \mathit{pre}^\sharp[a]^{\oplus}_{F^\sharp}(w^\sharp)(B) \tag{3}$$

$$\mathit{pre}^{+\oplus}_{F^\sharp}(w^\sharp)(B) = \max_{a \in A(B)} \mathit{pre}^\sharp[a]^{\oplus}_{F^\sharp}(w^\sharp)(B) \tag{4}$$

where $\oplus \in \{-,+\}$. These transformers are valid, i.e., $\mathit{pre}^{--}_{F^\sharp} \leq \alpha^- \circ \mathit{pre}^-_F \circ \gamma$, $\mathit{pre}^{++}_{F^\sharp} \geq \alpha^+ \circ \mathit{pre}^+_F \circ \gamma$, $\mathit{pre}^{-+}_{F^\sharp} \geq \alpha^+ \circ \mathit{pre}^-_F \circ \gamma$, and $\mathit{pre}^{+-}_{F^\sharp} \leq \alpha^- \circ \mathit{pre}^+_F \circ \gamma$, but not necessarily optimal, as we shall later.

To effectively compute these transformers, we now give a game construction whose valuation transformers equal exactly these transformers. Like in game-based abstraction, V_1 vertices are blocks, however, at a block B, player 1 chooses, from the enabled actions $A(B)$, an action a (rather than a set of abstract distributions). Player 2 chooses an abstract distribution from its successors. Here player 2 can choose from the abstractions of distributions that leave a state in block B with action a, which may include the zero distribution if a is not enabled on all states in the block. For a V_1 vertex v_1 where a is enabled, i.e.,

$a \in A(v_1)$, the successors of its (v_1, a) vertex (see Definition 2) represent the best transformers $pre^\sharp[a]^{\oplus}_{F^\sharp}$. Considering the size of compositional abstraction, the number $|V_2|$ of V_2 vertices is $|V_1| \times |A|$, in the worst case, compared to $2^{|V_p|}$ for the game-based abstraction of [18] (cf. Definition 1).

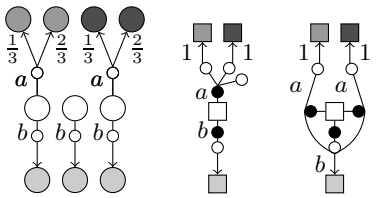
Definition 2. *Let $\mathcal{M} = (\Sigma, I, A, R)$ be an MDP, and let Q be a partition. The compositional game induced by Q is $\mathcal{G}^c_{\mathcal{M},Q} = ((V, E), V_{init}, (V_1, V_2, V_p), \delta)$, where $V_1 = Q$, $V_2 = \{(v_1, a) \mid a \in A(v_1)\} \subseteq V_1 \times A$, $V_p = \{\overline{\pi_{(s,a)}} \mid s \in S, a \in A\}$, $V_{init} = \{B \in Q \mid B \cap I \neq \emptyset\}$, and the edges E are defined by:*

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

$\delta : V_p \to Distr_V$ is the identity function.

Indeed, the valuation transformers defined in Equations (3) and (4) are exactly the valuation transformers of the game with respect to objective $F^\sharp$ of $\mathcal{G}^c$ from Definition 2.

Consider the MDP in Figure 3(a). Shades of gray show the state partition. In the resulting compositional game $\mathcal{G}^c$ in Figure 3(b), we focus on the white vertex (box in the middle). It has two V_2 vertices (solid circles): the a-vertex reflects that, on the lateral white states, action a leads to the northern blocks, while it is disabled on the central state. The latter induces a zero distribution (dangling vertex). The game-based abstraction in Figure 3(c) has three player 2 vertices reflecting the three different behaviors present in the white block. We now ask for minimal and maximal reachability probability to leave the white states. In the MDP, both are 1. In $\mathcal{G}^c$, we get probability interval $[0, 1]$ via action a, due to the zero distribution, and precisely probability 1 from action b. Overall this guarantees maximum $[1, 1]$, thanks to b, but only a minimum within $[0, 1]$, due to a. Game-based abstraction loses no precision and we get $[1, 1]$ in all cases.



(a) MDP (b) $\mathcal{G}^c$ (c) $\mathcal{G}$

Fig. 3. MDP and its abstractions.

While compositional abstraction is computed by abstracting each command separately, *sets* of simultaneously enabled commands [19] have to be considered for game-based abstraction. In a sequential program, commands do not overlap, and the two methods coincide. In concurrent programs, compositional abstraction sacrifices correlations between actions but gains speed. However, refinement then has to address resulting imprecision (e.g., isolating the central white state in Figure 3). If there are finitely many actions, any finite partition Q can be refined to a finite partition Q' on which compositional abstraction is at least as precise as game-based abstraction on Q. Hence compositional abstraction is complete with respect to game-based abstraction for the considered programs.

Limitations. The PRISM language [7] supports a module structure where a module contains different commands. Commands can either run independently, as assumed in our exposition, or synchronize with commands in other modules via actions. Our implementation eliminates synchronization by a model transformation which may increase the number of commands. [19] proposed an analysis for concurrent probabilistic programs using predicate abstraction which does not eliminate synchronization. Based on user selection, it applies game-based abstraction separately to modules (or clusters of modules). However it was not supplemented with refinement. Further, unlike our compositional abstraction, adding predicates to such an abstraction may never yield the precision of the best transformer. One way to fruitfully combine [19] with our approach, would perhaps be to abstract all commands separately, also the synchronizing ones, and to only use the model transformation if necessary to achieve full precision.

Constructing games by Predicate Abstraction. The compositional games are computed automatically from the program by SMT techniques. The construction works along the lines of [2] with a few additions. Thanks to the definition of the game, the construction also has the same complexity (in the number SMT solver calls), which is remarkable because the resulting abstract model admits derivation of both lower and upper bounds.

Referring to the full version of the paper for details, we here focus on notation needed to explain refinement. Let $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ be a probabilistic program, and let $\mathcal{M} = (\Sigma, I, A, R)$ be the semantics of $\mathtt{P}$. A predicate φ stands for the set of states satisfying it, i.e., $[\![\varphi]\!] \subseteq \Sigma$. A set of predicates $\mathcal{P}$ induces a finite partition of the state space: two states are in the same block if and only if they satisfy the same predicates. We denote the value of predicate φ in a block B by $B[\varphi_i]$, i.e., $[\![\varphi]\!]_s = B[\varphi]$ for all $s \in B$.

4.3 Refinement

The analysis begins with a few predicates able to express the reachability objective and refines the abstraction with fresh predicates until the difference between lower and upper bound is smaller than a pre-specified threshold ε for all initial vertices. To discover predicates, the refinement algorithm leverages the strategies that achieve the lower and upper bound respectively in the game. Overall the refinement method blends ideas from [2] and [3]. Like Probabilistic CEGAR [2], we consider the Markov chain induced by an optimal strategy, which makes it more likely that refinement affects the probabilities at initial states. Probabilistic CEGAR is based on checking a set of abstract paths enumerated by a shortest path algorithm, the refinement in the current paper makes no use of shortest-path enumeration, but instead refines guided by lower and upper bounds, similar to [3], however does so within an induced Markov chain, unlike [3]. Also the refinement strategy of [3] does not apply to our game structure.

Differences between lower and upper bound arise from non-determinism between distributions at player 2 vertices (different from [18, 3] where they appear at player 1 vertices). While in the concrete model, the choice of an action yields

a unique distribution, in the game, there may be multiple distributions with the same action, i.e., $|E(v_1, a)| > 1$ for $v_1 \in V_1$ and $a \in A(v_1)$. These account for states with different behavior that are contained in the same block. Splitting such block can gain precision, hence we say that a player 1 vertex $v_1 \in V_1$ is *refinable* if, for some action $a \in A(v_1)$, we have $|E(v_1, a)| > 1$.

The Algorithm RefBlock$(v_1, (v, v'))$, takes as input a pivot block $v_1 \in V_1$ (or player 1 vertex) to be refined, and two distinct distributions $v, v' \in E(v_1, a)$ with action a and returns a fresh predicates. To this end, the well-known concept of synthesizing predicates from weakest preconditions is used. Algorithm RefBlock works as follows:

If v or v' is the zero distribution, action a is enabled in some but not all states in block v_1. In this case, RefBlock returns the guard of command a and thus removes this zero distribution.

Otherwise, due to $v \neq v'$, there is an update $\mathtt{u}$ and states whose successors lie in B, and states with successor in B'. Recall that $B[\varphi]$ denotes the value of φ in B. Since $B \neq B'$, there is a predicate φ with $B[\varphi] \neq B'[\varphi]$. We add the weakest precondition $\mathtt{WP}_{\mathtt{u}}(\varphi)$ of predicate φ to the set of predicates, which describes exactly the set of states such that φ holds after executing update $\mathtt{u}$. To facilitate the discovery of such updates, distributions are, in fact, annotated with the respective updates inducing them. We omitted this detail for simplicity.

Refinement Algorithm. From the game, we compute valuations $w^{\leq} \in W_{V_1}$ and $w^{\geq} \in W_{V_1}$ for lower and upper bound respectively. For a player 1 vertex $v_1 \in V_1$, denote by $d(v_1) = w^{\geq}(v_1) - w^{\leq}(v_1)$ the difference between the two. Further optimal pure and memoryless player 1 and player 2 strategies are computed: $\sigma_1^{\leq}, \sigma_2^{\leq}$ for the lower bound and correspondingly $\sigma_1^{\geq}, \sigma_2^{\geq}$.

We choose a *pivot block* that lies within the MC induced by an optimal strategy, improving the chances that refinement affects the optimal probability of an initial vertex. We choose $v_1 \in V_1$ with maximal difference $d(v_1)$ according to the following lemma:

Lemma 2. *Assume $d(v) > 0$ for some initial vertex $v \in V_{init}$. Then there exists a refinable block reachable from v in the Markov chain induced by lower-bound strategies $\sigma_1 = \sigma_1^{\leq}$ and $\sigma_2 = \sigma_2^{\leq}$, in case of minimal probabilistic reachability, and upper-bound strategies $\sigma_1 = \sigma_1^{\geq}, \sigma_2 = \sigma_2^{\geq}$ otherwise.*

We can stop, if $d(v) \leq \varepsilon$ for all initial vertices $v \in V_{init}$ since we are only interested in computing the reachability probability of initial states. Otherwise, we leverage the strategies to localize where to refine the abstraction. Being pure and memoryless, the player 1 strategies determine an optimal player 2 vertex, and player 2 strategies determine a distribution: player 1 strategies yield vertices $\sigma_1^{\leq}(v_1) \in V_2$ and $\sigma_1^{\geq}(v_1) \in V_2$. Let $v_p^{\leq} = \sigma_2^{\leq}(\sigma_1^{\leq}(v_1))$ and $v_p^{\geq} = \sigma_2^{\geq}(\sigma_1^{\geq}(v_1))$ be the distributions determined by the player 2 strategies.

To compute predicates, RefBlock(v_1, p) is invoked on the pivot block v_1 and pairs p of distributions from two sets $R^{\leq}, R^{\geq} \subseteq V_p \times V_p$, which we explain below. We start with $R^{\leq}$, which aims at the lower bound: abstraction underestimates the lower bound by summarizing states corresponding to the "minimal

distribution" $v_p^{\leq}$ with states having a higher reachability probability. To separate them, $R^{\leq}$ contains pairs of distributions $(v_p^{\leq}, v^*)$ where $v^* \in E(\sigma_1^{\leq}(v_1))$ achieves a higher probability than $v_p^{\leq}$, i.e., $w^{\leq}(v_1) < \left(\sum_{v' \in V_1} v^*(v') \cdot w^{\leq}(v')\right)$. Dually, $R^{\geq}$ contains the pairs of distributions $(v_p^{\geq}, v^*)$ where $v^* \in E(\sigma_1^{\geq}(v_1))$ such that $w^{\geq}(v_1) > \left(\sum_{v' \in V_1} v^*(v') \cdot w^{\geq}(v')\right)$, in order to decrease the overestimation of the upper bound.

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 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`. Now we compare with `PASS 1.0` giving runtimes (in seconds) and results below:

	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	✔

✔ 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. More details like the parameters and number of refinement steps can be found in Appendix A. `PASS 2.0` and the case studies are available at `http://depend.cs.uni-sb.de/PASS/pass-cegar.html`.

6 Conclusion and Future Work

This paper provides a framework to construct abstract probabilistic analyses with optimal precision with respect to the best transformer. Based on this framework, we present an analysis for concurrent programs, which yields precise lower and upper probability bounds and is quite efficient. Orthogonal to this, our AI formalization also opens the way to effective approximations of the best transformer like [11]. We would like to extend our method with costs or rewards, such that valuations map to real numbers rather than just probabilities.

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. Ball, T., Podelski, A., Rajamani, S.K.: Boolean and Cartesian Abstraction for Model Checking C Programs. In: TACAS. (2001) 268–283
5. Graf, S., Saídi, H.: Construction of Abstract State Graphs with PVS. In: CAV. (1997) 72–83
6. Baier, C.: On Algorithmic Verification Methods for Probabilistic Systems (1998) Habilitationsschrift, Universität Mannheim.
7. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: TACAS. (2006) 441–444
8. 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
9. Cousot, P., Cousot, R.: Systematic Design of Program Transformation Frameworks by Abstract Interpretation. In: POPL '02, ACM (2002) 178–190
10. Alt, M., Ferdinand, C., Martin, F., Wilhelm, R.: Cache Behavior Prediction by Abstract Interpretation. In: SAS. (1996) 52–66
11. Ball, T., Podelski, A., Rajamani, S.K.: Boolean and Cartesian Abstraction for Model Checking C Programs. In: TACAS. (2001) 268–283
12. Reps, T.W., Sagiv, S., Yorsh, G.: Symbolic Implementation of the Best Transformer. In: VMCAI. (2004) 252–266
13. Monniaux, D.: Abstract interpretation of programs as markov decision processes. Sci. Comput. Program. **58** (2005) 179–205
14. Pierro, A.D., Wiklicky, H.: Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation. In: PPDP. (2000) 127–138
15. Smith, M.J.A.: Probabilistic Abstract Interpretation of Imperative Programs using Truncated Normal Distributions. ENTCS **220** (2008) 43–59
16. Coletta, A., Gori, R., Levi, F.: Approximating Probabilistic Behaviors of Biological Systems Using Abstract Interpretation. ENTCS **229** (2009) 165–182
17. Monniaux, D.: Backwards abstract interpretation of probabilistic programs. In: ESOP. (2001) 367–382
18. Kwiatkowska, M., Norman, G., Parker, D.: Game-based Abstraction for Markov Decision Processes. In: QEST. (2006) 157–166
19. Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Game-Based Probabilistic Predicate Abstraction in PRISM. In: QAPL. (2008)
20. Tarski, A.: A Lattice-Theoretical Fixpoint Theorem and Its Applications. In: Pacific Journal of Mathematics 5:2:. (1955) 285–309
21. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons (1994)
22. Condon, A.: The Complexity of Stochastic Games. Inf. Comput. **96** (1992) 203–224
23. Chatterjee, K., de Alfaro, L., Henzinger, T.A.: Trading Memory for Randomness. In: QEST. (2004) 206–217
24. Hinton, A., Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM: A Tool for Automatic Verification of Probabilistic Systems. In: TACAS. (2006) 441–444
25. Eppstein, D.: Finding the k Shortest Paths. SIAM J. Comput. **28** (1998) 652–673

Appendix A gives more details on experiments. Appendix B contains proofs for the different Propositions, Lemmas, Theorems and other claims made in the paper.

A Extended Experimental Results

PASS 2.0 features several optimizations. First, tranformers are generated on the fly. Starting with the initial vertices, we add their successors, and only compute abstract distributions for blocks transitively reachable from an initial vertex, further, before quantitative analysis, vertices that cannot reach the objective are removed. Though reachable in abstract, all states in a pivot block may be unreachable in the program. During refinement, we first check if a state in the pivot block is reachable before applying RefineBlock, otherwise we use interpolation to generate predicates from the spurious path, similar to [3]. PASS 2.0 interfaces with the MathSat SMT solver to check if an abstract path is spurious and generate interpolants in case of unsatisfiability. After a refinement, PASS 2.0 only recomputes the transformers of commands whose precision benefits from the new predicates.

Table 1. The first part of the table are model parameters and properties studied. On the right part, we display statistics: This table presents the results of the PRISM [24] tool hors concours. The number of states and transitions is given in thousands, i.e. 34K means 34,000.

Case study (parameters)			Property	PRISM states	PRISM trans	PRISM time	
WLAN BOFF/T	5	315	k=3	5,195K	11,377K	93	wlan1
	6	315	k=3	12,616K	28,137K	302	
	6	315	k=6	12,616K	28,137K	2024	wlan2
	6	9500	k=6	–	–	**TO**	
CSMA/CD BOFF	3		p1	41K	52K	10	
	4		p1	124K	161K	56	csma1
	3		p2	41K	52K	10	
	4		p2	124K	161K	21	csma2
BRP N/MAX	16	3	p1	2K	3K	5.4	
	32	5	p1	5K	7K	12	brp1
	64	5	p1	10K	14K	26	
	>16	3	p4	∞	–	–	
	>16	4	p4	∞	–	–	brp2
	>16	5	p4	∞	–	–	
SW			goodput	∞	–	–	sw1
			timeout	∞	–	–	sw2
Zeroconf				∞	–	–	zeroconf
Herman				∞	–	–	herman

We have considered several case studies using a Linux PC with Pentium^{TM} 4 2.6 GHz and 1.0 GB RAM. Table 1 shows statistics about the models (if available), the mapping of the model names in the short description to the particular instance of the respective model, and the results of PRISM hors concours if applicable (model is finite).

Table 2. The first part of the table are model parameters and properties studied. On the right part, we display statistics for PASS [2], and PASS 2.0. Apart from reachable state numbers, number of transitions (non-zero entries in transition matrix), and computation time (in seconds), the number of iterations of the refinement loop (refs), and of predicates generated (preds). The number of states and transitions is given in thousands, i.e. 34K means 34,000.

Case study			PASS						PASS 2.0				
(parameters)		Property	states	trans	refs	preds	paths	time	states	trans	refs	preds	time
WLAN BOFF/T	5 315	k=3	34K	36K	9	120	604	72	5K	3K	4	107	43
	6 315	k=3	34K	42K	9	116	604	88	5K	3K	4	110	57
	6 315	k=6	771K	113K	9	182	582	306	68K	32K	4	140	115
	6 9500	k=6	771K	113K	9	182	582	311	68K	32K	3	138	76
CSMA/CD BOFF	3	p1	1K	2K	8	58	28	9	1K	1K	4	73	7
	4	p1	6K	9K	14	100	56	38	3K	4K	5	85	10
	3	p2	0.5K	0.9K	12	41	28	10	148	159	2	60	5
	4	p2	0.5K	1.5K	12	41	44	11	188	201	2	56	5
BRP N/MAX	16 3	p1	2K	3K	9	46	41	9	2K	2K	19	51	8
	32 5	p1	5K	7K	9	64	111	21	5K	5K	36	68	27
	64 5	p1	10K	14K	8	95	585	91	10K	10K	69	102	76
	>16 3	p4	0.5K	0.9K	7	26	17	3	861	1K	3	34	1
	>16 4	p4	0.6K	1K	7	27	17	4	1K	1K	4	35	1
	>16 5	p4	0.7K	1K	8	28	18	5	1K	2K	5	36	2
SW		goodput	5K	11K	3	40	7	87	890	1K	8	44	18
		timeout	27K	44K	3	49	6	89	831	1K	5	30	2

Table 2 shows for each case study model sizes, analysis time in seconds, and refinement steps. The timeout limit is two hours (timeout is indicated by **TO**). The finite-state models have been analyzed with PRISM 3.1.1, PASS and PASS 2.0. PASS is the implementation of the abstraction refinement method from [2]. Most models are taken from the paper [2]. Some models have an infinite state space denoted by "∞" state space size. These were analyzed with PASS and PASS 2.0.

PASS 2.0 computes lower and upper bounds for minimal and maximal reachability and stops once the relative difference gets below 10^{-4}. The abstraction of [2] computes upper bounds for maximal probabilistic reachability. PRISM computes precise probabilities if applicable. Now we discuss the analysis results.

The IEEE 802.11 *WLAN* protocol uses a randomized algorithm to avoid collisions on the transmission channel. Further, the wireless medium loses messages with a certain probability. The considered model is parametrized with an *exponential* back-off counter limit BOFF and a maximal package send time of T μs. We checked the property: "The maximum probability that either station's back-off counter reaches k" for k=3 and k=6. As shown in Table 2, increasing BOFF from 5 to 6 leads to an exponential increase in model size and running time in PRISM, while in PASS the row is identical for BOFF=5 and BOFF=6. It was observed in [2] that states with back-off counter higher than three can reach a goal state (via a reset), however they do not lie on paths with maximal probability. Hence refinement never splits blocks with respect to back-offs beyond three which is somewhat similar to the motivating example. The discrepancy in the number of blocks between PASS and PASS 2.0 is caused by the optimizations mentioned at the beginning. In addition, we used PASS 2.0 to compute the mini-

mal reachability probability for reaching a backoff counter value of 6. PASS does not support minimal reachability.

Similar to WLAN, the IEEE 802.3 *CSMA/CD* protocol is parametrized with an *exponential* back-off counter limit BOFF. We analyzed the properties: (p1): "The maximum probability that both stations deliver", and (p2): "The message of any station eventually delivered before 1 backoff". For both properties, as shown in the table, the abstract state space is significantly smaller. Consider property p2. Similar to the WLAN protocol, the size of the abstraction does not change with respect to the size of BOFF. However, the counterexample analysis of PASS explores a larger number of paths explored with increasing BOFF. The reason is that for greater values of BOFF, there is more branching in the probabilistic model, thus in the abstraction there are more abstract paths being explored. Refinement for probabilistic game abstraction explores one path per refinement step. Therefore this problem does not arise.

The *Bounded Retransmission Protocol* (BRP) protocol has two parameters: N denotes the length of the file to be transmitted, and MAX denotes the maximal number of retransmissions. We have studied "Property 1" and "Property 4" (p1 and p4 in the table). As observed in [2], there is little leeway for abstraction for p1 if precise probabilities are required. The modeled has to be fully reconstructed by refinement. However, p4 can be analyzed for an infinite parameter range with PASS: it is an invariant property with respect to the file length N. The constraint $N > 16$ allows us to verify the property for any possible file length greater than 16. While PASS computed safe upper bounds, we are able to compute the precise probability. We can thus confirm that the bounds previously computed are tight.

The *Sliding Window Protocol* (SW) is the standard protocol with lossy channels over an unbounded domain of sequence numbers. Due to the sequence numbers, the model is infinite and hence not amenable to PRISM. Two properties were checked. A goodput property which considers the difference between the number of sent and received packages. The goal is to compute probability that the number of sent packages exceeds the number of received packages by a particular constant. PASS established that, at any time, the probability of the difference exceeding three is at most three percent for windows size four. PASS 2.0 computed the precise probability, which is actually smaller than three percent. The second property concerns the probability of a protocol timeout. The number of blocks used by the game-based tool is significantly smaller than those computed by PASS. This is because PASS, unlike PASS 2.0, unnecessarily unrolled the value domain of a timer variable.

Discussion Based on the experiments described in Section 5, we can give an elaborate comparison. Probabilistic CEGAR checks if the maximal reachability probability is below a user-specified threshold value. If not, a Markov chain is computed from the abstract model which is an abstract counterexample against the specification. From the Markov chain, paths are enumerated starting with the paths with highest measure using a k-shortest path algorithm [25]. The paths are checked for feasibility with respect to the original program. Feasible paths give a lower bound. Infeasible paths provide predicates. The shortest path algorithm

is costly for a large abstract state space, e.g. for the WLAN model. The analysis of many paths requires a lot of calls to the SMT solver. In contrast, probabilistic game abstraction only analyzes one path per refinement step. The path is found by a cheap breadth first search. Sometimes `PASS` explores sets of paths that correspond to multiple unrollings of loops in the abstract automaton. It thus generates predicates which are discovered by probabilistic game abstraction after multiple iterations steps. Therefore, `PASS` takes fewer iterations in some cases. However, the strength of the probabilistic game abstraction is that it better focuses on the parts of the abstract model that are imprecise. This often leads to more succinct abstractions. Further, our experiments show that the probabilistic game abstraction is computed efficiently with roughly the same speed as [2].

B Proofs and Remarks

B.1 Background on AI

In the 50s, Tarski gave a characterization of least and greatest fixpoints [20]:

Theorem 2. *Let $f : L \to L$ be a monotone function over a lattice $(L, \leq)$. Then Tarski's theorem guarantees that $Fix(f) = \{x \in L \mid f(x) = x\}$ is a lattice and that least and greatest fixpoint exist, and are given by*

$$lfp_{\leq}(f) = \bigsqcap Fix(f) = \bigsqcap \{x \in L | f(x) \leq x\}$$
$$gfp_{\leq}(f) = \bigsqcup Fix(f) = \bigsqcup \{x \in L | f(x) \geq x\} \ .$$

We recall this theorem because it guarantees existence of several fixpoints used in the paper, and we later use it in proofs.

Best Transformer. We want to show that the best transformer is the most precise valid transformer, as claimed in Section 2, which is a well-known fact. We prove it to keep this work self-contained. We recall the terminology in that section. For transformer $f : L \to L$, the *best transformer* [9], is the composition of functions:

$$f^\sharp = \alpha \circ f \circ \gamma \ .$$

We show that $f^\sharp$ is the most precise abstract transformer that is a valid abstraction of f, i.e., $f^\sharp \leq g^\sharp$ for any valid transformer $g^\sharp : M \to M$.

Let $m \in M$. Validity follows by choosing $a_1 = (f \circ \gamma)(m)$, $b_1 = f^\sharp(w^\sharp)$, and exploiting that $\alpha(a_1) = b_1$ implies $(f \circ \gamma)(m) = a_1 \leq \gamma(b_1) = (\gamma \circ f^\sharp)(m)$. Optimality follows by setting $a_2 = f \circ \gamma(m)$, $b_2 = g^\sharp(m)$, because $g^\sharp$ is a valid transformer, we have $a_2 \leq \gamma(b_2)$ which implies $f^\sharp(m) = \alpha \circ f \circ \gamma(m) = \alpha(a_2) \leq b_2 = g^\sharp(m)$.

B.2 Markov Decision Processes (Section 2.2)

Infinitely Branching MDPs. Concerning the definition of MDPs, it is sufficient for most purposes to assume that MDPs are finitely branching, i.e. the set of actions is finite $|A| < \infty$. This covers both the programs considered in [3] and concurrent probabilistic programs as considered in Section 4.1.

In presence of infinitely branching MDPs, several complications arise. Simple schedulers are no longer optimal. The definition of probabilistic reachability in terms of valuation transformers has to be changed, since the valuation transformers employ minima and maxima over a possibly infinite set, which might not exist. Further, the game-based abstraction of an infinitely branching MDP for a given finite partition could be infinite.

Summarizing, infinitely branching MDPs still admit the notion of best transformers presented (even if their game-based abstraction is infinite). To this end, one needs to re-define the valuation transformers to use infima and suprema (rather than minima and maxima). The latter are guaranteed to exist since the interval $[0, 1]$ is compact.

B.3 Proposition 1

Proof. We want to establish the following two Galois connections:

(a) $(W_\Sigma, \geq) \underset{\alpha^-_\gamma}{\overset{\gamma}{\leftrightarrows}} (W_Q, \geq)$ (lower-bound abstraction)

(b) $(W_\Sigma, \leq) \underset{\alpha^+}{\overset{\gamma}{\leftrightarrows}} (W_Q, \leq)$ (upper-bound abstraction)

Monotonicity follows directly by definition. We now show that the requirements of a Galois connection are met. We begin with (a): the lower-bound abstraction. We need to show that for all $w \in W_\Sigma$ and $w^\sharp \in W_Q$, we have $w^\sharp \leq \alpha^-(w) \Leftrightarrow \gamma(w^\sharp) \leq w$. We first show "$\Rightarrow$": we assume $w^\sharp \leq \alpha^-(w)$. Therefore, for all $B \in Q$, $w^\sharp(B) \leq \alpha^-(w)(B) = \min_{s \in B} w(s)$. For all $s \in \Sigma$, we have $\gamma(w^\sharp)(s) = w^\sharp(\overline{s}) \leq \min_{s \in B} w(s) \leq w(s)$. Now we show "$\Leftarrow$". We assume that, for all $w \in W_\Sigma, w^\sharp \in W_Q$ and $s \in \Sigma$, we have $\gamma(w^\sharp)(s) \leq w(s)$. Let $w \in W_\Sigma, w^\sharp \in W_Q$ and $B \in Q$. Then $w^\sharp(B) = \min_{s \in B} \gamma(w^\sharp)(s) \leq \min_{s \in B} w(s) = \alpha^-(w)(B)$.

We proceed with (b): the upper-bound abstraction. We show that for all $w \in W_\Sigma$ and $w^\sharp \in W_Q$, we have $\alpha^+(w) \leq w^\sharp \Leftrightarrow w \leq \gamma(w^\sharp)$. We first show "$\Rightarrow$": we assume $\alpha^+(w) \leq w^\sharp$. Therefore, for all $B \in Q$, $\max_{s \in B} w(s) = \alpha^+(w)(B) \leq w^\sharp(B)$. For all $s \in \Sigma$, we have $w(s) \leq \max_{s \in B} w(s) = \gamma(w^\sharp)(s) = w^\sharp(\overline{s})$. Now we show "$\Leftarrow$". We assume that, for all $w \in W_\Sigma, w^\sharp \in W_Q$ and $s \in \Sigma$, we have $w(s) \leq \gamma(w^\sharp)(s)$. Let $w \in W_\Sigma, w^\sharp \in W_Q$ and $B \in Q$. The claim then follows by $\alpha^+(w)(B) = \max_{s \in B} w(s) \leq \max_{s \in B} \gamma(w^\sharp)(s) = w^\sharp(B)$.

The lower-bound abstraction α^- is a $\bigsqcap$-morphism because:

$$(\alpha^-(\bigsqcap M))(B) = \bigsqcap_{s \in B} \bigsqcap_{w \in M} w(s) = \left(\bigsqcap_{w \in M} \alpha^-(w) \right)(B)$$

for all $M \subseteq W_\Sigma$ and all blocks $B \in Q$.

The upper-bound abstraction α^+ is a $\bigsqcap$-morphism, since:

$$(\alpha^+(\bigsqcup M))(B) = \bigsqcup_{s\in B} \bigsqcup_{w\in M} w(s) = \left(\bigsqcup_{w\in M} \alpha^+(w)\right)(B)$$

for all $M \subseteq W_\Sigma$ and all blocks $B \in Q$.

The concretization function is a morphism with respect to both operators:

$$(\gamma(\bigsqcup M^\sharp))(s) = \left(\bigsqcup_{w\in M^\sharp} \gamma(w^\sharp)\right)(s)$$

$$(\gamma(\bigsqcap M^\sharp))(s) = \left(\bigsqcap_{w\in M^\sharp} \gamma(w^\sharp)\right)(s)$$

for all $M^\sharp \subseteq W_Q$ and all states $s \in \Sigma$.

■

B.4 Lemma 1

We show a slightly more general theorem from which the claim follows immediately by applying the fixpoint characterization of probabilistic reachability. We show that: Fixpoints of the best transformers yield lower and upper bounds on the least fixpoint of a given valuation transformer:

$$\gamma(gfp_{\geq}(\alpha^- \circ f \circ \gamma)) \leq lfp_{\leq}(f) \leq \gamma(lfp_{\leq}(\alpha^+ \circ f \circ \gamma)) \ .$$

where $f : W_\Sigma \to W_\Sigma$.

Proof. In the proof, we only exploit properties of best transformers, i.e. we do not leverage any specifics of valuations. The result itself and the proof techniques are standard for AI: combining properties of the Galois connection with Tarski's fixpoint Theorem. Nevertheless, we give a complete proof.

We let $f^+ = (\alpha^+ \circ f \circ \gamma)$. Abstract transformer f^+ is a valid abstraction of f with respect to $\leq$, i.e., $f \circ \gamma \leq \gamma \circ f^+$. Therefore, we have:

$$f(\gamma(lfp_{\leq}(f^+))) \leq (\gamma \circ f^+)(lfp_{\leq}(f^+)) = \gamma(lfp_{\leq}(f^+))$$

Therefore, $\gamma(lfp_{\leq}(f^+)) \leq \bigsqcap\{w \in W_\Sigma \mid f(w) \leq w\} = lfp_{\leq}(f)$.

We let $f^- = (\alpha^- \circ f \circ \gamma)$. Abstract transformer f^- is a valid abstraction of f with respect to $\geq$, i.e., $f \circ \gamma \geq \gamma \circ f^-$. From the properties of the Galois connection, it follows that:

$$(\alpha^- \circ f) \leq (\alpha^- \circ f) \circ (\gamma \circ \alpha^-) = (\alpha^- \circ f \circ \gamma) \circ \alpha^- = (f^- \circ \alpha^-)$$

From this observation, we get the follow disequality:

$$f^-(\alpha^-(lfp_{\leq}(f))) \leq (\alpha^- \circ f)(lfp_{\leq}(f)) = \alpha^-(lfp_{\leq}(f))$$

Hence

$$\begin{aligned} gfp_{\geq}(f^-) &= lfp_{\leq}(f^-) \\ &= \bigsqcap\{w^\sharp \in W_Q \mid f^-(w^\sharp) \leq w^\sharp\} \\ &\leq \alpha^-(lfp_{\leq}(f)) \end{aligned}$$

and therefore, since (α^-, γ) is a Galois connection, we get the equivalent statement $\gamma(gfp_{\geq}(f^-))) \leq lfp_{\leq}(f)$, which proves the claim. ■

B.5 Theorem 1

Proof. It follows, by $\gamma(w^\sharp)(s') = w^\sharp(\overline{s}')$ and from the defintion of $\overline{\pi}$, that, for $w^\sharp \in W_Q$, for a state $s \in \Sigma$ and a block in $B \in Q$ such that $s \in B$, $\pi \in Distr(s)$, $\pi^\sharp = \overline{\pi}$, we have:

$$\sum_{s' \in \Sigma} \pi(s') \cdot (\gamma(w^\sharp))(s') = \sum_{B' \in Q} \pi(B') \cdot w^\sharp(B') = \sum_{B' \in Q} \pi^\sharp(B') \cdot w^\sharp(B') \quad (5)$$

For $D \subseteq Distr_\Sigma$, we define $\overline{D} = \{\overline{\pi} \mid \pi \in D\}$.

We call two states $s, s' \in \Sigma$, behavorially equivalent under abstraction, denoted by $s \equiv_Q^{\sim} s'$, iff the abstraction of the steps functions is equal:

$$s \equiv_Q^{\sim} s' \Leftrightarrow \overline{Distr(s)} = \overline{Distr(s')} \ . \quad (6)$$

We denote the equivalence class of a state s with respect to this equivalence by $[s]_Q^{\sim}$ and denote the quotient by $\Sigma / \equiv_Q^{\sim}$. We shall drop the superscript and the subscript if the equivalence is clear from context and write $[s]$ instead. The equivalence is also an equivalence on each subset of Σ for example on blocks. We use the analogous notation $B / \equiv_Q^{\sim}$.

From the definition, we have that, for a player 1 vertex $v_1 \in V_1$:

$$E(v_1) = \{\overline{Distr(s)} \mid [s] \in v_1 / \equiv_Q^{\sim}\} \quad (7)$$

Now we prove $pre_F^{-+} = \alpha^- \circ pre^+ \circ \gamma$, i.e. we have to prove for every valuation $w^\sharp \in W_Q$ and every block $v \in V_1$ that $pre_F^{-+}(w^\sharp)(v) = \alpha^- \circ pre^+ \circ \gamma(w^\sharp)(v)$. Let $w^\sharp \in W_Q$ and $B \in Q$. If $B = F$ or $B = F_0$ respectively, the claim is trivially

fulfilled by definition. Let us assume that $B \in Q \setminus \{F, F_0\}$. Then we have:

$$\begin{aligned}
pre_F^{-+}(w^\sharp)(v) &\overset{\text{def.}}{=} \min_{v_2 \in E(v)} \max_{v_p \in v_2} \sum_{v' \in V_1} \delta(v_p)(v') \cdot w^\sharp(v') \\
&\overset{(7)}{=} \inf_{[s] \in v/\equiv_Q^{\rightsquigarrow}} \max_{\pi^\sharp \in \overline{Distr(s)}} \sum_{B' \in Q} \pi^\sharp(v') \cdot w^\sharp(B') \\
&\overset{(6)}{=} \inf_{s \in v} \max_{\pi^\sharp \in \overline{Distr(s)}} \sum_{B' \in Q} \pi^\sharp(v') \cdot w^\sharp(B') \\
&\overset{(5)}{=} \inf_{s \in v} \max_{\pi \in Distr(s)} \sum_{s' \in \Sigma} \pi(s') \cdot w^\sharp(\overline{s}') \\
&= ((\alpha^- \circ pre_F^+ \circ \gamma)(w^\sharp))(v)
\end{aligned}$$

The other cases pre_F^{--}, pre_F^{+-} and pre_F^{++} respectively follow in total analogy to the above proof. Therefore, we skip them. ■

B.6 Compositional Abstraction (Section 4.2)

We recall the proof obligations:

(a) We need to show that Equations 1 and 2 characterize the valuation transformers induced by the compositional game.
(b) The valuation transformers induced by the compositional game are valid.
(c) Compositional abstraction is complete w.r.t. game-based abstraction.

Proof. We begin with part (a). If $B \in F^\sharp$, the claim is trivially fulfilled, similarly if $F^\sharp$ is not reachable from B. We start with upper bounds for minimal and maximal reachability. The first step is establish a characterization of the best transformer of an action. Let $v_1 \in V_1$, $a \in A(v_1)$, $v_2 = (v_1, a)$ and $w^\sharp \in W_Q$. Then we have:

$$\begin{aligned}
\max_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') &= \sup_{[s] \in v_1/\equiv_Q^{\rightsquigarrow}} \sum_{B' \in Q} \overline{\pi_{(s,a)}}(v') \cdot w^\sharp(B') \\
&= \sup_{s \in v} \sum_{s' \in \Sigma} \pi_{(s,a)}(s') \cdot w^\sharp(\overline{s}') \\
&= pre^\sharp[a]^+_{F^\sharp}(w^\sharp)(v_1)
\end{aligned}$$

This implies that for $v_1 \in V_1$ and $w^\sharp \in W_Q$ we have:

$$\begin{aligned}
pre_{F^\sharp}^{-+}(w^\sharp)(v_1) &= \min_{v_2 \in E(v_1)} \max_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') \\
&= \min_{a \in A(B)} pre^\sharp[a]^+_{F^\sharp}(w^\sharp)(v_1) \\
&= \min_{a \in A(B)} pre^\sharp[a]^+_{F^\sharp}(w^\sharp)(v_1)
\end{aligned}$$

and

$$\begin{aligned} pre^{++}_{F^\sharp}(w^\sharp)(v_1) &= \max_{v_2 \in E(v_1)} \max_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') \\ &= \max_{a \in A(B)} pre^\sharp[a]^+_{F^\sharp}(w^\sharp)(v_1) \ . \end{aligned}$$

We continue with lower bounds for minimal and maximal reachability. Again, the first step is establish a characterization of the best transformer of an action. Let $v_1 \in V_1$, $a \in A(v_1)$, $v_2 = (v_1, a)$ and $w^\sharp \in W_Q$. Then we have:

$$\begin{aligned} \min_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') &= \inf_{[s] \in v_1/\equiv_{\overrightarrow{Q}}} \sum_{B' \in Q} \overline{\pi_{(s,a)}}(v') \cdot w^\sharp(B') \\ &= \inf_{s \in v} \sum_{s' \in \Sigma} \pi_{(s,a)}(s') \cdot w^\sharp(\overline{s}') \\ &= pre^\sharp[a]^-_{F^\sharp}(w^\sharp)(v_1) \end{aligned}$$

This implies that for $v_1 \in V_1$ and $w^\sharp \in W_Q$ we have:

$$\begin{aligned} pre^{--}_{F^\sharp}(w^\sharp)(v_1) &= \min_{v_2 \in E(v_1)} \min_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') \\ &= \min_{a \in A(B)} pre^\sharp[a]^-_{F^\sharp}(w^\sharp)(v_1) \\ &= \min_{a \in A(B)} pre^\sharp[a]^-_{F^\sharp}(w^\sharp)(v_1) \end{aligned}$$

and

$$\begin{aligned} pre^{+-}_{F^\sharp}(w^\sharp)(v_1) &= \max_{v_2 \in E(v_1)} \min_{v_p \in E(v_2)} \sum_{v' \in V_1} \delta(v_p) \cdot w^\sharp(v') \\ &= \max_{a \in A(B)} pre^\sharp[a]^-_{F^\sharp}(w^\sharp)(v_1) \ . \end{aligned}$$

We show part (b), namely that the obtained valuation transformers are valid abstractions and we establish the relation to best transformers.

We have for all valuations $w \in W_\Sigma$ and all states $s \in \Sigma$:

$$pre^-_F(w)(s) = \min_{a \in A(s)} pre[a]_F(w)(s) \tag{8}$$

$$pre^+_F(w)(s) = \max_{a \in A(s)} pre[a]_F(w)(s) \ . \tag{9}$$

We show that $\gamma \circ pre^{+-}_{F^\sharp} \leq pre^+_F \circ \gamma \leq \gamma \circ pre^{++}_{F^\sharp}$. For all valuations $w^\sharp \in W_Q$ and all blocks $B \in Q$, we have

$$\begin{aligned} \left((\gamma \circ pre^{+-}_{F^\sharp})(w^\sharp)\right)(B) &\overset{(9)}{=} \max_{a \in A(B)} (\gamma \circ pre^\sharp[a]^-_{F^\sharp})(w^\sharp)(B) \\ &\leq \max_{a \in A(B)} (pre[a]_F \circ \gamma)(w^\sharp)(B) \\ &\leq \left((pre^+_F \circ \gamma)(w^\sharp)\right)(B) \end{aligned}$$

and we also have

$$\begin{aligned}\left((\gamma \circ pre^{++}_{F^\sharp})(w^\sharp)\right)(B) &\overset{(9)}{=} \max_{a \in A(B)} (\gamma \circ pre^\sharp[a]^+_{F^\sharp})(w^\sharp)(B) \\ &\geq \max_{a \in A(B)} (pre[a]_F \circ \gamma)(w^\sharp)(B) \\ &\geq \left((pre^+_F \circ \gamma)(w^\sharp)\right)(B)\ .\end{aligned}$$

The proof for $\gamma \circ pre^{--}_{F^\sharp} \leq pre^-_F \circ \gamma \leq \gamma \circ pre^{-+}_{F^\sharp}$ works analogous to the proof for maximum reachability. Therefore, we skip this part.

Finally part (c) follows by employing the quotient construction in the proof of Theorem 1. For a given finite quotient, this defines a finite partition on which compositional abstraction is at least as precise as game-based abstraction. ■

B.7 Proof of Lemma 2

Proof. We prove the claim by contradiction. Let us consider minimal reachability, as the other case works analogously. Assume that $d(v) > 0$ for an initial vertex v. Further, let us assume that there exists no such refinable block reachable via the lower-bound strategies. The lower-bound player 1 strategy assigns an optimal player 2 vertex to each player 1 vertex. By assumption, these player 2 vertices have at most one successor. Hence the lower bound and the upper bound agree on the player 1 vertices reachable from v via the lower-bound strategies. Since this lower-bound strategy determines the lower bound, we have $d(v) = 0$. Contradiction. ■

C Details on Predicate Abstraction

We give more detail on the construction of games using predicate abstraction mentioned in subsection 4.2.

Let $\mathtt{P} = (\mathtt{X}, \mathtt{I}, \mathtt{C})$ be a probabilistic program, and let $\mathcal{M} = (\Sigma, I, A, R)$ be the semantics of $\mathtt{P}$. A predicate φ stands for the set of states satisfying it, namely $[\![\varphi]\!] \subseteq \Sigma$. We fix a set of predicates $\mathcal{P} = \{\varphi_1, ..., \varphi_n\}$. The set $\mathcal{P}$ induces a finite partition of the set of states: two states are in the same block if and only if they satisfy the same predicates. A block is uniquely represented by a bit vector $(b_0, ...b_n) \in \{0,1\}^n$ where $B = \{s \mid b_i = [\![\varphi_i]\!]_s\}$. We denote, for predicate φ_i, the corresponding truth value b_i in $bv(B)$ by $B[\varphi_i]$. The abstract game $\mathcal{G}_{\mathcal{M},\mathcal{P}}$ induced by $\mathcal{P}$ is constructed as follows:

The player 1 vertices V_1 are the state partitions. player 2 vertices are tuples consisting of a player 1 vertex and an action from the set $A(v_1)$ of the player 1 vertex v_1. Beside the sets $A(v_1)$, it is also of interest whether a block v_1 contains states on which some action from $A(v_1)$ are not enabled. To this end, we also compute the set $A^{-1}(v_1) = \cap_{s \in B} A(s)$ which contains the set of actions that are enabled on all states in block B.

To obtain $A(v_1)$ and $A^{-1}(v_1)$, we first compute, for each command, the set of blocks $Q^-(a)$ containing only states on which the command is enabled: $Q^-(a) =$

$\{B \mid B \subseteq [\![\mathtt{g}_a]\!]\}$. Further, we compute the set of blocks containing a state on which the command is enabled: $Q^+(a) = \{B \mid B \cap [\![\mathtt{g}_a]\!] \neq \emptyset\}$. Q^- is the most precise under-approximation and Q^+ the most precise over-approximation of $\mathtt{g}_a$ expressible in terms of minterms over Q. The following equalities hold: $A^-(B) = \{a \mid B \in Q^-(a)\}$ and $A(B) = \{a \mid B \in Q^+(a)\}$. Computing $A(B)$ and $A^-(B)$ is the only added cost compared to [2].

The edges from a player 1 vertex v_1 to player 2 vertices (v_1, a) are determined by the sets $A(v_1)$ whose computation is described above. To obtain the out-going edges of a player 2 vertex (v_1, a) and the corresponding distributions, we need to compute the set $\{\mathit{lift}_Q(\pi_{(s,a)}) \mid s \in v_1\}$. An algorithm to compute the set $\{\mathit{lift}_Q(\pi_{(s,a)}) \mid s \in v_1, a \in A(s)\}$ is described in [2]. We use the same algorithm as in [2], and then add the zero distribution if and only if $A(v_1) \neq A^-(v_1)$.