Form Methods Syst Des (2010) 36: 246–280
DOI 10.1007/s10703-010-0097-6

A game-based abstraction-refinement framework for Markov decision processes

Mark Kattenbelt · Marta Kwiatkowska · Gethin Norman · David Parker

Published online: 3 August 2010

Abstract In the field of model checking, abstraction refinement has proved to be an extremely successful methodology for combating the state-space explosion problem. However, little practical progress has been made in the setting of probabilistic verification. In this paper we present a novel abstraction-refinement framework for Markov decision processes (MDPs), which are widely used for modelling and verifying systems that exhibit both probabilistic and nondeterministic behaviour. Our framework comprises an abstraction approach based on stochastic two-player games, two refinement methods and an efficient algorithm for an abstraction-refinement loop. The key idea behind the abstraction approach is to maintain a separation between nondeterminism present in the original MDP and nondeterminism introduced during the abstraction process, each type being represented by a different player in the game. Crucially, this allows lower and upper bounds to be computed for the values of reachability properties of the MDP. These give a quantitative measure of the quality of the abstraction and form the basis of the corresponding refinement methods. We describe a prototype implementation of our framework and present experimental results demonstrating automatic generation of compact, yet precise, abstractions for a large selection of real-world case studies.

Keywords Probabilistic verification · Markov decision processes · Abstraction · Abstraction refinement

M. Kattenbelt · M. Kwiatkowska · D. Parker (✉)
Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford OX1 3QD, UK
e-mail: david.parker@comlab.ox.ac.uk

M. Kattenbelt
e-mail: mark.kattenbelt@comlab.ox.ac.uk

M. Kwiatkowska
e-mail: marta.kwiatkowska@comlab.ox.ac.uk

G. Norman
Department of Computing Science, University of Glasgow, 18 Lilybank Gardens, Glasgow G12 8RZ, Scotland
e-mail: gethin@dcs.gla.ac.uk

1 Introduction

Numerous real-life systems from a wide range of application domains, including communication and network protocols, security protocols and distributed algorithms, exhibit both *probabilistic* and *nondeterministic* behaviour, and therefore developing techniques to verify the correctness of such systems is an important research topic. Markov decision processes (MDPs) are a natural and widely used model for this purpose and the automatic verification of MDPs using *probabilistic model checking* has proved successful for their analysis. Despite improvements in implementations and tool support in this area, the state-space explosion problem remains a major hurdle for the practical application of these methods.

This paper is motivated by the success of *abstraction-refinement* techniques [8], which have been established as one of the most effective ways of attacking the state-space explosion problem in non-probabilistic model checking. The basic idea is to construct a smaller *abstract model*, by removing details from the concrete system not relevant to the property of interest, which is consequently easier to analyse. This is done in such a way that when the property is verified true in the abstraction it also holds in the concrete system. On the other hand, if the property does not hold in the abstraction, information from the model checking process (typically a counterexample) is used either to show that the property is false in the concrete system or to *refine* the abstraction. This process forms the basis of a loop which refines the abstraction until the property is shown to be true or false in the concrete system.

In the probabilistic setting, it is typically necessary to consider *quantitative* properties, in which case the actual probability or expectation of some event must be determined, e.g. "the probability of reaching an error state within T time units" or "the expected power consumption before termination". We therefore adopt a quantitative approach when defining a correspondence between properties of the concrete and abstract models. One example would be the case where quantitative results computed from the abstraction constitute conservative bounds on the actual values for the concrete model. In fact, due to the presence of nondeterminism in an MDP there is not necessarily a single value corresponding to a given quantitative measure. Instead, *best-case* and *worst-case* scenarios are analysed. More specifically, model checking of MDPs typically reduces to computation of *probabilistic reachability* and *expected reachability* properties, namely the minimum or maximum probability of reaching a set of states, and the minimum or maximum expected reward or cost incurred when reaching a set of states.

When constructing an abstraction of an MDP, the resulting model will invariably exhibit a greater degree of nondeterminism caused by the uncertainty with regards to the precise behaviour of the system that the abstraction process introduces. The key idea in our abstraction approach is to maintain a distinction between the nondeterminism from the original MDP and the nondeterminism introduced during the abstraction process. To achieve this, we model abstractions of MDPs as stochastic two-player games [9, 40], where the two players correspond to the two different forms of nondeterminism. We can then analyse these models using techniques developed for such games [5, 10, 15].

Our analysis of these abstract models results in a separate lower and upper bound for each of the minimum and maximum probabilities (or expected rewards) of reaching a set of states. This approach is particularly appealing since this information provides both a quantitative measure of the quality (or preciseness) of the abstraction and an indication of how to improve it. By comparison, if no discrimination between the two forms of nondeterminism is made, a single lower and upper bound would be obtained. Therefore, in the (common) situation where the minimum and maximum probabilities (or expected rewards) are notably different, the quantitative measure and corresponding basis for refinement would be lost.

Consider, for example, the extreme case where the two-player game approach reveals that the minimum probability of reaching some set of states is in the interval $[0, \varepsilon_{\min}]$ and the maximum probability is in the interval $[1-\varepsilon_{\max}, 1]$. In this case, a single pair of bounds could at best establish that both the minimum and maximum probability lie within the interval $[0, 1]$, effectively yielding no information about the utility of the abstraction or how to refine it in order to improve the bounds.

Based on the separate bounds obtained through this approach, we present two methods for automatically refining our game-based abstractions of MDPs and an efficient algorithm for an abstraction-refinement loop. In addition, using a prototype implementation of the framework, we give experimental results that demonstrate automatic generation of compact, yet precise, abstractions for a large selection of MDP case studies. For convenience, the current prototype performs model-level abstractions, first building an MDP in the probabilistic model checker PRISM [22, 37] and then reducing it to a stochastic two-player game. Our framework is also designed to function with higher-level abstraction methods such as predicate abstraction [19], which has been adapted to probabilistic models [25, 44]. In fact, the techniques presented in this paper have subsequently been applied to build abstractions of probabilistic models using both predicate abstraction [26] and convex polyhedra [30].

A preliminary version of this paper, introducing the game-based abstraction approach, was published in conference proceedings as [28].

Outline of the paper In the next section, we present background material on Markov decision processes and stochastic two-player games required in the remainder of the paper. Section 3 describes our abstraction technique and shows its correctness. Based on this, Sect. 4 introduces a corresponding abstraction-refinement framework. In Sect. 5, we describe a prototype implementation of the techniques from Sects. 3 and 4, and present experimental results for a large selection of real-world case studies. Section 6 discusses related work and Sect. 7 concludes the paper.

2 Background

Let $\mathbb{R}_{\geq 0}$ denote the set of non-negative reals. For a finite set Q, we denote by $\mathsf{Dist}(Q)$ the set of *discrete probability distributions* over Q, i.e. the set of functions $\mu : Q \to [0, 1]$ such that $\sum_{q \in Q} \mu(q) = 1$.

2.1 Probability measures and spaces

We begin by briefly introducing some relevant notions from probability and measure theory. For further details, the interested reader is referred to, for example, [3, 27].

Definition 1 Let Ω be an arbitrary non-empty set and $\mathcal{F}$ a family of subsets of Ω. We say that $\mathcal{F}$ is a *σ-field* on Ω if:

1. the empty set $\emptyset$ is in $\mathcal{F}$;
2. whenever A is an element of $\mathcal{F}$, the complement $\Omega \backslash A$ is also in $\mathcal{F}$;
3. whenever A_i is an element of $\mathcal{F}$ for each $i \in \mathbb{N}$, the union $\bigcup_{i \in \mathbb{N}} A_i$ is also in $\mathcal{F}$.

The elements of $\mathcal{F}$ are called *measurable sets*, and $(\Omega, \mathcal{F})$ is called a *measurable space*.

Definition 2 (Probability measure and space) Let $(\Omega, \mathcal{F})$ be a measurable space. A function $Prob : \mathcal{F} \to [0, 1]$ is a *probability measure* on $(\Omega, \mathcal{F})$ and $(\Omega, \mathcal{F}, Prob)$ a *probability space*, if the following conditions hold:

1. $Prob(\Omega) = 1$
2. if $A_1, A_2, \ldots$ is a disjoint sequence of elements of $\mathcal{F}$, then $Prob(\bigcup_i A_i) = \sum_i Prob(A_i)$.

For any family $\mathcal{A}$ of subsets of Ω, there exists a unique smallest σ-field containing $\mathcal{A}$ [27], which we call the σ-field *generated* by $\mathcal{A}$. Furthermore, given that $\mathcal{A}$ satisfies certain properties (see e.g. [27] for details), and we define the probability for each element of $\mathcal{A}$ with a function $Prob : \mathcal{A} \to [0, 1]$, then the function is guaranteed to extend uniquely to a probability measure on the σ-field generated by $\mathcal{A}$.

2.2 Discrete-time Markov chains

We now introduce discrete-time Markov chains (DTMCs), a model for systems with only probabilistic behaviour. A DTMC can be thought of as a state transition system, where transitions are annotated with probabilities indicating the likelihood of their occurrence.

Definition 3 A *DTMC* is a tuple $\mathsf{D} = (S, s_{init}, \mathbf{P})$ where:

- S is a set of states;
- $s_{init} \in S$ is an initial state;
- $\mathbf{P} : S \times S \to [0, 1]$ is a *transition probability matrix*, with $\sum_{s' \in S} \mathbf{P}(s, s') = 1$ for all $s \in S$.

For any states $s, s' \in S$, the transition probability matrix gives the probability $\mathbf{P}(s, s')$ of making a transition from s to s'. A *path* of a DTMC represents an execution of the system that it is modelling. Formally, a path of a DTMC is a non-empty, finite or infinite sequence of states $\pi = s_0 s_1 s_2 \ldots$ such that $\mathbf{P}(s_i, s_{i+1}) > 0$ for all $i \geq 0$. We denote by $\pi(i)$ the state s_i. For a finite path π_{fin}, we let $|\pi_{fin}|$ denote the length of π_{fin} (i.e. number of transitions), and let $last(\pi_{fin})$ be its final state. Finally, $\pi^{(i)}$ denotes the *prefix* of length i of π, i.e. the finite path $\pi(0) \ldots \pi(i)$. The sets of all finite and infinite paths starting in state s are denoted $Path_{fin}(s)$ and $Path(s)$, respectively.

To reason about the probabilistic behaviour of the DTMC, we need to determine the probability that certain paths are taken. This is achieved by defining, for each state $s \in S$, a probability space $(Path(s), \mathcal{F}_s, Prob_s)$ over the (infinite) paths from s. Below, we give an outline of this construction. For further details, see [27].

The probability measure $Prob_s$ is induced by the transition probability matrix $\mathbf{P}$ as follows. First, for any finite path $\pi_{fin} \in Path_{fin}(s)$ of length n, we define the probability $\mathbf{P}_s(\pi_{fin})$:

$$\mathbf{P}_s(\pi_{fin}) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } n = 0 \\ \mathbf{P}(\pi_{fin}(0), \pi_{fin}(1)) \cdots \mathbf{P}(\pi_{fin}(n-1), \pi_{fin}(n)) & \text{otherwise.} \end{cases}$$

Next, we define the *cylinder set* of π_{fin}, denoted $cyl(\pi_{fin})$, as follows:

$$cyl(\pi_{fin}) \stackrel{\text{def}}{=} \{\pi \in Path(s) \mid \pi_{fin} \text{ is a prefix of } \pi\}$$

i.e. the cylinder set $cyl(\pi_{fin})$ is the set of all infinite paths which have π_{fin} as a prefix. We denote by $Cyl(s)$ the family of cylinder sets $cyl(\pi_{fin})$, where π_{fin} ranges over the elements of $Path_{fin}(s)$. Then, let $\mathcal{F}_s$ be the smallest σ-field on $Path(s)$ that contains the family

of sets $Cyl(s)$. Due to properties of $Cyl(s)$, we can then define the probability measure on $\mathcal{F}_s$ as the unique measure which extends the function $Prob_s : Cyl(s) \to [0,1]$ where $Prob_s(cyl(\pi_{fin})) = \mathbf{P}_s(\pi_{fin})$ for all $\pi_{fin} \in Path_{fin}(s)$.

2.3 Markov decision processes

Markov decision processes (MDPs) are a natural representation for the modelling and analysis of systems with both probabilistic and nondeterministic behaviour.

Definition 4 A *Markov decision process* is a tuple $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$, where

- S is a set of states;
- $s_{init} \in S$ is an initial state;
- $Steps : S \to 2^{\mathsf{Dist}(S)}$ is a *probabilistic transition function*;
- $\mathbf{r} : S \times \mathsf{Dist}(S) \to \mathbb{R}_{\geq 0}$ is a *reward function*.

A probabilistic transition $s \xrightarrow{\mu} s'$ is made from a state s by first nondeterministically selecting a distribution $\mu \in Steps(s)$ and then making a probabilistic choice of target state s' according to the distribution μ. As for a DTMC, a *path* of an MDP corresponds to an execution of the system that it is modelling. It represents a particular resolution of both nondeterminism *and* probability. Formally, a path of an MDP is a non-empty, finite or infinite sequence of probabilistic transitions:

$$\pi = s_0 \xrightarrow{\mu_0} s_1 \xrightarrow{\mu_1} s_2 \xrightarrow{\mu_2} \cdots$$

such that $\mu_i(s_{i+1}) > 0$ for all $i \geq 0$. We denote by $\pi(i)$ the state s_i and by $step(\pi, i)$ the distribution μ_i. We extend the notions of length, final state and prefix of paths from DTMCs to MDPs in the standard manner.

In contrast to a path, an *adversary* (sometimes also known as a *policy*, *strategy* or *scheduler*) represents a particular resolution of nondeterminism *only*. An adversary resolves the nondeterministic choice in a state of an MDP, based on the history of its execution up to that point. More precisely, an adversary is a function mapping every finite path π_{fin} to a distribution $\mu \in Steps(last(\pi_{fin}))$. For any state $s \in S$ and adversary A, let $Path^A_{fin}(s)$ and $Path^A(s)$ denote the sets of finite and infinite paths starting in s that correspond to A.

Definition 5 An adversary A is called *memoryless* (or *simple*) if, for any finite paths π_{fin} and π'_{fin} for which $last(\pi_{fin}) = last(\pi'_{fin})$, we have $A(\pi_{fin}) = A(\pi'_{fin})$.

The behaviour of an MDP from a state s, under a given adversary A, is purely probabilistic and can be viewed as a DTMC whose states are given by the set of finite paths $Path^A_{fin}(s)$. Formally, the behaviour can be described by the infinite-state DTMC $\mathsf{D}^A_s = (S^A_s, s, \mathbf{P}^A_s)$, where:

- $S^A_s = Path^A_{fin}(s)$;
- for any two finite paths $\pi_{fin}, \pi'_{fin} \in S^A_s$ we have

$$\mathbf{P}^A_s(\pi_{fin}, \pi'_{fin}) = \begin{cases} \mu(s') & \text{if } \pi'_{fin} \text{ is of the form } \pi_{fin} \xrightarrow{a,\mu} s' \text{ and } A(\pi_{fin}) = (a, \mu) \\ 0 & \text{otherwise.} \end{cases}$$

There is a one-to-one correspondence between the infinite paths of the DTMC D_s^A and $Path^A(s)$. It therefore follows that, using the probability measure construction for DTMC given in Sect. 2.2, we can construct a probability space $(Path^A(s), \mathcal{F}_s^A, Prob_s^A)$. Based on this construction, we now introduce two quantitative measures for MDPs which together form the basis for probabilistic model checking of MDPs [1, 13].

The first measure is *probabilistic reachability*, which refers to the probability of reaching a set of target states. For adversary A, the probability of reaching the target set $F \subseteq S$ from state s is given by:

$$p_s^A(F) \stackrel{\text{def}}{=} Prob_s^A\{\pi \in Path^A(s) \mid \exists i \in \mathbb{N}.\pi(i) \in F\}.$$

The second measure we consider is *expected reachability*, which refers to the expected reward accumulated before reaching a set of target states. For an adversary A, the expected reward of reaching the target set F from state s, denoted $e_s^A(F)$, is defined as the expectation [3], with respect to the probability measure $Prob_s^A$, of the random variable $r(F,\cdot) : Path^A(s) \to \mathbb{R}$ which returns, for a given path, the total reward accumulated until a state in F is reached along the path. Formally we have:

$$e_s^A(F) \stackrel{\text{def}}{=} \int_{\pi \in Path^A(s)} r(F,\pi)\,\mathrm{d}Prob_s^A$$

where for any path $\pi \in Path^A(s)$:

$$r(F,\pi) \stackrel{\text{def}}{=} \begin{cases} \sum_{i=1}^{\min\{j \mid \pi(j) \in F\}} \mathbf{r}(\pi(i-1), step(\pi, i-1)) & \text{if } \exists j \in \mathbb{N}.\pi(j) \in F \\ \infty & \text{otherwise.} \end{cases}$$

For simplicity, we have defined the reward of a path that does not reach F to be ∞, even though the total reward of the path may not be infinite. Essentially, this means that the expected reward of reaching F from s under A is finite if and only if, under the adversary A, a state in F is reached from s with probability 1.

A natural way to reason about the behaviour of an MDP is to consider the *minimum* and *maximum* values of these measures, quantifying over all adversaries.

Definition 6 For an MDP $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$, set of target states $F \subseteq S$ and state $s \in S$, the *minimum and maximum reachability probabilities* of reaching F from s equal:

$$p_s^{\min}(F) = \inf_A p_s^A(F) \quad \text{and} \quad p_s^{\max}(F) = \sup_A p_s^A(F)$$

and the *minimum and maximum expected rewards* of reaching F from s equal:

$$e_s^{\min}(F) = \inf_A e_s^A(F) \quad \text{and} \quad e_s^{\max}(F) = \sup_A e_s^A(F).$$

We write, for example, $\mathbf{p}^{\min}(F)$ to denote the vector of values $p_s^{\min}(F)$ over all states $s \in S$. More generally, for any vector $\mathbf{x}$ over S, we write x_s to denote the element for state $s \in S$.

Computing values for probabilistic and expected reachability reduces to the *stochastic shortest path problem* for Markov decision processes; see for example [2, 12]. A key result in this respect is that optimality with respect to probabilistic and expected reachability can always be achieved with memoryless adversaries (see Definition 5). A consequence of this

is that these quantities can be computed through a method known as *value iteration* [2, 12], which iteratively computes increasingly precise approximations to the required values. The lemma below forms the basis for value iteration.

Lemma 1 *Consider an MDP* $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$ *and set of target states* $F \subseteq S$.

- *The sequences of vectors* $\langle \mathbf{p}^{\min,(n)} \rangle_{n\in\mathbb{N}}$ *and* $\langle \mathbf{p}^{\max,(n)} \rangle_{n\in\mathbb{N}}$ *converge to the vectors* $\mathbf{p}^{\min}(F)$ *and* $\mathbf{p}^{\max}(F)$, *where for any state* $s \in S$:
  - *if* $s \in F$, *then* $p_s^{\min,(n)} = p_s^{\max,(n)} = 1$ *for all* $n \in \mathbb{N}$;
  - *if* $s \notin F$, *then*:

$$p_s^{\min,(n)} = \begin{cases} 0 & \text{if } n = 0 \\ \min_{\mu\in Steps(s)} \sum_{s'\in S} \mu(s') \cdot p_{s'}^{\min,(n-1)} & \text{otherwise} \end{cases}$$

$$p_s^{\max,(n)} = \begin{cases} 0 & \text{if } n = 0 \\ \max_{\mu\in Steps(s)} \sum_{s'\in S} \mu(s') \cdot p_{s'}^{\max,(n-1)} & \text{otherwise.} \end{cases}$$

- *The sequences of vectors* $\langle \mathbf{e}^{\min,(n)} \rangle_{n\in\mathbb{N}}$ *and* $\langle \mathbf{e}^{\max,(n)} \rangle_{n\in\mathbb{N}}$ *converge to the vectors* $\mathbf{e}^{\min}(F)$ *and* $\mathbf{e}^{\max}(F)$, *where for any state* $s \in S$:
  - *if* $s \in F$, *then* $e_s^{\min,(n)} = e_s^{\max,(n)} = 0$ *for all* $n \in \mathbb{N}$;
  - *if* $s \notin F$, *then*:

$$e_s^{\min,(n)} = \begin{cases} \infty & \text{if } p_s^{\max} < 1 \\ 0 & \text{if } p_s^{\max} = 1 \text{ and } n = 0 \\ \min_{\mu\in Steps(s)}(\mathbf{r}(s,\mu) + \sum_{s'\in S} \mu(s') \cdot e_{s'}^{\min,(n-1)}) & \text{otherwise} \end{cases}$$

$$e_s^{\max,(n)} = \begin{cases} \infty & \text{if } p_s^{\min} < 1 \\ 0 & \text{if } p_s^{\min} = 1 \text{ and } n = 0 \\ \max_{\mu\in Steps(s)}(\mathbf{r}(s,\mu) + \sum_{s'\in S} \mu(s') \cdot e_{s'}^{\max,(n-1)}) & \text{otherwise.} \end{cases}$$

In practice, value iteration is carried out by computing the vector of required values for increasingly large values of n, until a pre-specified convergence criterion is met. One common approach is to check that the maximum difference between the corresponding elements of successive vectors is below some fixed threshold δ. Another is to use the maximum relative difference of vector elements.

2.4 Stochastic two-player games

Finally in this section, we describe (simple) stochastic games [9, 40], which are turn-based games involving two players and chance.

Definition 7 A *stochastic two-player game* is a tuple $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_p), \delta, \overline{\mathbf{r}})$ where:

- (V, E) is a finite directed graph;
- $v_{init} \in V$ is an initial vertex;
- (V_1, V_2, V_p) is a partition of V;
- $\delta : V_p \to \mathsf{Dist}(V)$ is a *probabilistic transition function*;
- $\overline{\mathbf{r}} : E \to \mathbb{R}_{\geq 0}$ is a *reward function* over edges.

Vertices in V_1, V_2 and V_p are called 'player 1', 'player 2' and 'probabilistic' vertices, respectively. The game proceeds in steps, moving from a vertex v to one of its neighbouring vertices v' in the game graph. The choice of v' depends on the type of the vertex v. If $v \in V_1$ then player 1 chooses v', if $v \in V_2$ then player 2 makes the choice, and if $v \in V_p$ then v' is selected randomly according to the distribution $\delta(v)$. A Markov decision process can thus be thought of as a stochastic game in which there are no player 2 vertices and where there is a strict alternation between player 1 and probabilistic vertices.

A *play* in a game G is a sequence of vertices $\omega = v_0 v_1 v_2 \ldots$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. We denote by $\omega(i)$ the vertex v_i and, for a finite play ω_{fin}, we write $last(\omega_{fin})$ for the final vertex of ω_{fin} and $|\omega_{fin}|$ for its length (the number of transitions). The prefix of length i of play ω is denoted $\omega^{(i)}$.

A *strategy* for player 1 is a function $\sigma_1 : V^* V_1 \to \mathsf{Dist}(V)$, i.e. a function from the set of finite plays ending in a player 1 vertex to the set of distributions over vertices, such that for any $\omega_{fin} \in V^* V_1$ and $v \in V$, if $\sigma_1(\omega_{fin})(v) > 0$, then $(last(\omega_{fin}), v) \in E$. Strategies for player 2, denoted by σ_2, are defined analogously. For a fixed pair of strategies (σ_1, σ_2) we denote by $Play_{fin}^{\sigma_1,\sigma_2}(v)$ and $Play^{\sigma_1,\sigma_2}(v)$ the set of finite and infinite plays starting in vertex v that correspond to these strategies. For strategy pair (σ_1, σ_2), the behaviour of the game is completely random and, for any vertex v, we can construct a probability space $(Play^{\sigma_1,\sigma_2}(v), \mathcal{F}_v^{\sigma_1,\sigma_2}, Prob_v^{\sigma_1,\sigma_2})$. This construction proceeds in similar fashion to the one described for MDPs in the previous section.

We can then, as for MDPs, define the notions of *probabilistic reachability* and *expected reachability*. For a fixed strategy pair (σ_1, σ_2), vertex $v \in V$ and a set of target vertices $\overline{F} \subseteq V$, we define the corresponding probability of reaching $\overline{F}$ as:

$$p_v^{\sigma_1,\sigma_2}(\overline{F}) \stackrel{\text{def}}{=} Prob_v^{\sigma_1,\sigma_2}\{\omega \in Play^{\sigma_1,\sigma_2}(v) \mid \exists i \in \mathbb{N} \wedge \omega(i) \in \overline{F}\}$$

and the expected reward of reaching $\overline{F}$ as:

$$e_v^{\sigma_1,\sigma_2}(\overline{F}) \stackrel{\text{def}}{=} \int_{\omega \in Play^{\sigma_1,\sigma_2}(v)} \overline{r}(\overline{F}, \omega)\, \mathrm{d}Prob_v^{\sigma_1,\sigma_2}$$

where for any play $\omega \in Play^{\sigma_1,\sigma_2}(v)$:

$$\overline{r}(\overline{F}, \omega) \stackrel{\text{def}}{=} \begin{cases} \sum_{i=1}^{\min\{j \mid \omega(j) \in \overline{F}\}} \overline{\mathbf{r}}(\omega(i-1), \omega(i)) & \text{if } \exists j \in \mathbb{N}.\omega(j) \in \overline{F} \\ \infty & \text{otherwise.} \end{cases}$$

Definition 8 For a game $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_p), \delta, \overline{\mathbf{r}})$, set of target vertices $\overline{F} \subseteq V$ and vertex $v \in V$, the optimal probabilities of the game for player 1 and player 2, with respect to $\overline{F}$ and v, are defined as follows:

$$\sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(\overline{F}) \quad \text{and} \quad \sup_{\sigma_2} \inf_{\sigma_1} p_v^{\sigma_1,\sigma_2}(\overline{F})$$

and the optimal expected rewards are:

$$\sup_{\sigma_1} \inf_{\sigma_2} e_v^{\sigma_1,\sigma_2}(\overline{F}) \quad \text{and} \quad \sup_{\sigma_2} \inf_{\sigma_1} e_v^{\sigma_1,\sigma_2}(\overline{F}).$$

A player 1 strategy σ_1 is optimal from vertex v with respect to the probability of reaching target set $\overline{F}$ if:

$$\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(\overline{F}) = \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(\overline{F}). \tag{1}$$

The optimal strategies for player 2 and for expected rewards can be defined analogously. We now summarise a number of results from [5, 9, 10].

Definition 9 A strategy σ_i is *pure* if it does not use randomisation, that is, for any finite play ω_{fin} such that $last(\omega_{fin}) \in V_i$, there exists $v' \in V$ such that $\sigma_i(\omega_{fin})(v') = 1$. A strategy σ_i is *memoryless* if its choice depends only on the current vertex, that is, $\sigma_i(\omega_{fin}) = \sigma_i(\omega'_{fin})$ for any finite plays ω_{fin} and ω'_{fin} such that $last(\omega_{fin}) = last(\omega'_{fin})$.

As is the case with MDPs, for any stochastic game, the family of pure memoryless strategies suffices for optimality with respect to probabilistic and expected reachability. Moreover, these values can be computed through an iterative processes known as value iteration [10].

Lemma 2 *Consider a stochastic game* $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$ *and set of target vertices* $\overline{F} \subseteq V$.

- *The sequence of vectors* $\langle \mathbf{p}^{(n)} \rangle_{n \in \mathbb{N}}$ *converges to the optimal probabilities for player* 1 *with respect to the target set* $\overline{F}$, *where for any vertex* $v \in V$:
  - *if* $v \in \overline{F}$, *then* $p_v^{(n)} = 1$ *for all* $n \in \mathbb{N}$;
  - *and otherwise*:

$$p_v^{(n)} = \begin{cases} 0 & \text{if } n = 0 \\ \max_{(v,v') \in E} p_{v'}^{(n-1)} & \text{if } n > 0 \text{ and } v \in V_1 \\ \min_{(v,v') \in E} p_{v'}^{(n-1)} & \text{if } n > 0 \text{ and } v \in V_2 \\ \sum_{v' \in V} \delta(v)(v') \cdot p_{v'}^{(n-1)} & \text{if } n > 0 \text{ and } v \in V_\mathrm{p}. \end{cases}$$

- *The sequence of vectors* $\langle \mathbf{e}^{(n)} \rangle_{n \in \mathbb{N}}$ *converges to the optimal expected rewards for player* 1 *with respect to the target set* $\overline{F}$, *where for any vertex* $v \in V$:
  - *if* $v \in \overline{F}$, *then* $e_v^{(n)} = 0$ *for all* $n \in \mathbb{N}$;
  - *if* $\sup_{\sigma_2} \inf_{\sigma_1} p_v^{\sigma_1,\sigma_2}(\overline{F}) < 1$, *then* $e_v^{(n)} = \infty$ *for all* $n \in \mathbb{N}$;
  - *and otherwise*:

$$e_v^{(n)} = \begin{cases} 0 & \text{if } n = 0 \\ \max_{(v,v') \in E} (\overline{\mathbf{r}}(v, v') + e_{v'}^{(n-1)}) & \text{if } n > 0 \text{ and } v \in V_1 \\ \min_{(v,v') \in E} (\overline{\mathbf{r}}(v, v') + e_{v'}^{(n-1)}) & \text{if } n > 0 \text{ and } v \in V_2 \\ \sum_{v' \in V} (\overline{\mathbf{r}}(v, v') + \delta(v)(v') \cdot e_{v'}^{(n-1)}) & \text{if } n > 0 \text{ and } v \in V_\mathrm{p}. \end{cases}$$

Lemma 2 forms the basis of an iterative method to compute the vector of optimal values for a game. Note that, although this concerns only the optimal probability for player 1, similar results hold for player 2. Observe the similarity between this and the value iteration method for MDP solution described in Lemma 1.

3 Abstraction for Markov decision processes

We now present our notion of *abstraction* for MDPs, which is based on stochastic two-player games. The key idea is to maintain a distinction between the nondeterminism in an MDP and the nondeterminism introduced by abstracting it. The abstraction of an MDP is a stochastic two-player game in which the choices made by player 1 correspond to the nondeterminism from the process of abstraction and the choices made by player 2 correspond to the nondeterminism in the original MDP.

For an MDP $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$, which we refer to as the *concrete model*, we construct an abstraction based on a partition $P = \{S_1, S_2, \ldots, S_n\}$ of its state space S. The elements of P are referred to as *abstract states*, each comprising a set of *concrete states*. Intuitively, in the stochastic game representing the abstraction, player 1 vertices are abstract states and player 2 vertices correspond to the sets of nondeterministic choices in individual concrete states.

In order to formally define the abstraction process, we need a few preliminary definitions. For any distribution μ over S, we denote by $\overline{\mu}$ the probability distribution over P *lifted* from μ, i.e. $\overline{\mu}(S_i) = \sum_{s \in S_i} \mu(s)$ for all $S_i \in P$. In addition, for any state s, we denote by $\overline{\mathit{Steps}(s)}$ the set of reward-distribution pairs $\{(\mathbf{r}(s,\mu), \overline{\mu}) \mid \mu \in \mathit{Steps}(s)\}$. The inclusion of rewards in the definition of $\overline{\mathit{Steps}(s)}$ ensures that the reward values associated with probability distributions in the concrete model are preserved, even in the case where multiple (concrete) distributions are lifted to the same distribution over abstract states. This differs from [28] where, to simplify the presentation, we assumed that distributions that were lifted to the same abstract distribution had identical reward values.

Definition 10 Given an MDP $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$ and a partition $P = \{S_1, \ldots, S_n\}$ of the state space S, we define the corresponding abstraction $\mathsf{G}^P_{\mathsf{M}}$ as the stochastic game:

$$\mathsf{G}^P_{\mathsf{M}} = ((V, E), v_{init}, (V_1, V_2, V_{\mathrm{p}}), \delta, \overline{\mathbf{r}})$$

in which:

- $V_1 = P$;
- $V_2 = \{v_2 \subseteq \mathbb{R}_{\geq 0} \times \mathsf{Dist}(P) \mid v_2 = \overline{\mathit{Steps}(s)}$ for some $s \in S\}$;
- $V_{\mathrm{p}} = \{v_{\mathrm{p}} \in \mathbb{R}_{\geq 0} \times \mathsf{Dist}(P) \mid v_{\mathrm{p}} \in \overline{\mathit{Steps}(s)}$ for some $s \in S\}$;
- $v_{init} = S_i$ where $s_{init} \in S_i$;
- $(v, v') \in E$ if and only if one of the following conditions holds:
  - $v \in V_1$, $v' \in V_2$ and $v' = \overline{\mathit{Steps}(s)}$ for some $s \in v$;
  - $v \in V_2$, $v' \in V_{\mathrm{p}}$ and $v' \in v$;
  - $v \in V_{\mathrm{p}}$, $v' \in V_1$ and $v = (r, \overline{\mu})$ such that $\overline{\mu}(v') > 0$;
- $\delta : V_{\mathrm{p}} \to \mathsf{Dist}(V)$ projects $(r, \overline{\mu}) \in V_{\mathrm{p}}$ onto $\overline{\mu}$;
- $\overline{\mathbf{r}} : E \to \mathbb{R}_{\geq 0}$ is the reward function such that for any $(v, v') \in V \times V$:

$$\overline{\mathbf{r}}(v, v') = \begin{cases} r & \text{if } (v, v') \in V_2 \times V_{\mathrm{p}}, \text{ and } v' = (r, \overline{\mu}) \text{ for some } \overline{\mu} \in \mathsf{Dist}(P) \\ 0 & \text{otherwise.} \end{cases}$$

Example 1 We illustrate the abstraction process on a simple example, shown in Fig. 1. The concrete model (the MDP) is given in Fig. 1(a): states are drawn as circles and probability distributions as labelled arrows, grouped with arcs. For clarity, we omit rewards in this case. We use the partition $P = \{\{s_0\}, \{s_1, s_2, s_3\}, \{s_4, s_5\}, \{s_6\}\}$. The abstraction process is illustrated in two steps. Firstly, Fig. 1(b) shows the partition P, as dotted lines around groups of states, and the result of lifting each distribution μ in the MDP to $\overline{\mu}$. Secondly, Fig. 1(c) shows the stochastic game representing the corresponding abstraction. Player 1 vertices are drawn as grey rectangles, each corresponding to an element of the partition P. The player 2 vertices that are successors of each player 1 vertex are drawn as white shapes, enclosed within it. The concrete states corresponding to each player 2 vertex are also depicted inside the player 2 vertices. These are states from the MDP whose outgoing choices are identical after lifting. Probabilistic vertices are not drawn explicitly, but rather shown as groups of probabilistic transitions emanating from each player 2 vertex, in the same style as the MDP.

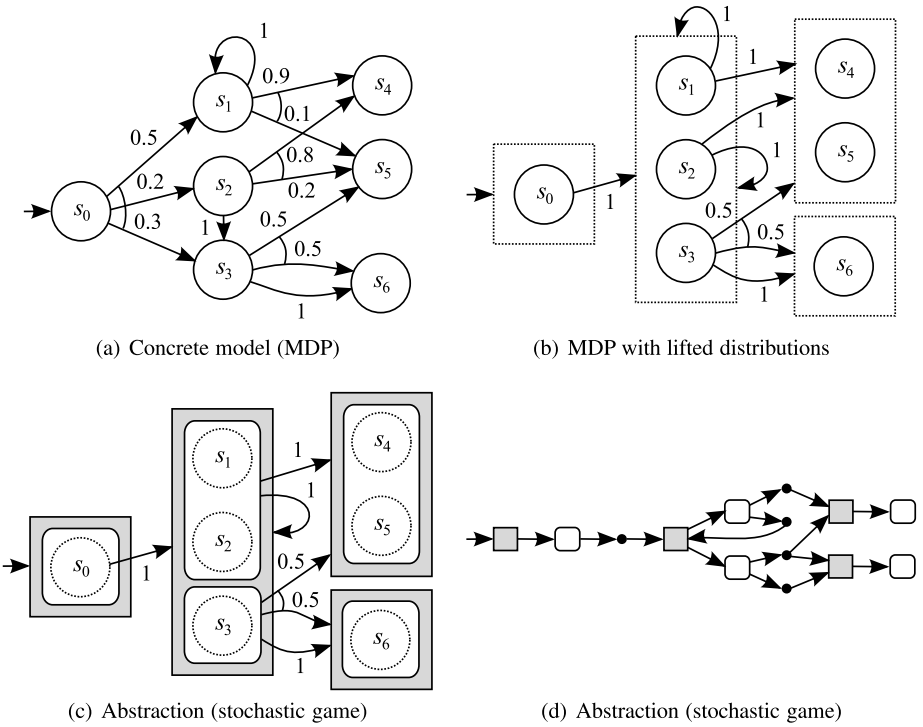


Fig. 1 Simple example of the abstraction process for Markov decision processes

For completeness, we also show, in Fig. 1(d), the same stochastic game with all vertices illustrated explicitly. Player 1, player 2 and probabilistic vertices are coloured grey, white and black, respectively.

Intuitively, the roles of the vertices and players in the abstraction can be understood as follows. A player 1 vertex corresponds to an abstract state: an element of the partition P of the states from the original MDP. Player 1 chooses a concrete state from this set and then player 2 chooses a probability distribution from those available in the concrete state (which is now a distribution over abstract states rather than concrete states).

This description perhaps misleadingly gives the impression that the stochastic game representing the abstraction is no smaller than the original concrete model. This is not the case. Firstly, note that V_2 vertices are actually sets of reward-distribution pairs that correspond to concrete states, not the concrete states themselves. Hence, concrete states with the same reward values and outgoing distributions (after lifting) are collapsed into one player 2 vertex (see, for example, s_1 and s_2 in Fig. 1(c)). Furthermore, in practice there is no need to store the entire vertex set V of the stochastic game. Since we have a strict alternation between V_1, V_2 and V_p vertices, we need only store the vertices in V_1, the outgoing transitions comprising each reward-distribution pair from V_1, and how these pairs are grouped (into elements of V_2). Later, in Example 3 and Sect. 5 we will show how, on all the case studies considered, the abstraction process brings a significant reduction in model size.

3.1 Analysis of the abstraction

In this section we describe how, for an MDP M and partition P, the stochastic game $\mathsf{G}^P_{\mathsf{M}}$ representing the abstraction of M yields lower and upper bounds for probabilistic reachability and expected reachability properties of M. To ease notation we let, for $x \in \{p, e\}$ and $\overline{F} \subseteq V$:

$$x_v^{lb,\min}(\overline{F}) \stackrel{\text{def}}{=} \inf_{\sigma_1,\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$x_v^{ub,\min}(\overline{F}) \stackrel{\text{def}}{=} \sup_{\sigma_1} \inf_{\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$x_v^{lb,\max}(\overline{F}) \stackrel{\text{def}}{=} \sup_{\sigma_2} \inf_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$x_v^{ub,\max}(\overline{F}) \stackrel{\text{def}}{=} \sup_{\sigma_1,\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F}).$$

The values $x_v^{ub,\min}(\overline{F})$ and $x_v^{lb,\max}(\overline{F})$ are the optimal reachability probabilities (if $x = p$) and expected rewards (if $x = e$) for players 1 and 2, respectively. As we have seen in Lemma 2, these optimal values can be generated using a simple iterative computation. This process can also be used to construct a pair of (memoryless) strategies that result in these optimal values [10]. The other values, $x_v^{lb,\min}(\overline{F})$ and $x_v^{ub,\max}(\overline{F})$, although not usually considered for stochastic games (because the players co-operate), can be computed similarly by considering the game as an MDP [2] (see Lemma 1). The following theorem shows how these four values represent bounds on the corresponding properties of the MDP.

Theorem 1 *Let $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$ be an MDP, P a partition of S and $F \in P$ a set of target states. If $\mathsf{G}^P_{\mathsf{M}} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$ is the stochastic game constructed from M and P according to Definition 10, then for any $x \in \{e, p\}$ and $s \in S$:*

$$x_v^{lb,\min}(\overline{F}) \leq x_s^{\min}(F) \leq x_v^{ub,\min}(\overline{F})$$
$$x_v^{lb,\max}(\overline{F}) \leq x_s^{\max}(F) \leq x_v^{ub,\max}(\overline{F})$$

where v is the unique vertex of V_1 such that $s \in v$ and $\overline{F} = \{F\}$.

We also show that using a *finer* partition results in a *more precise* abstraction, i.e. a stochastic game providing tighter lower and upper bounds.

Definition 11 If P and P' are partitions of a state space S, then P is *finer* than P' (denoted $P \preceq P'$) if, for any $v \in P$, there exists $v' \in P'$ such that $v \subseteq v'$. Equivalently, we say that P' is *coarser* than P.

Theorem 2 *Let $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$ be an MDP, P and P' partitions of S such that $P \preceq P'$ and $F \in P \cap P'$ a set of target states. If $\mathsf{G}^P_{\mathsf{M}} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$ and $\mathsf{G}^{P'}_{\mathsf{M}} = ((V', E'), v'_{init}, (V'_1, V'_2, V_\mathrm{p}'), \delta', \overline{\mathbf{r}}')$ are the stochastic games constructed from P and P' according to Definition 10, then for any $x \in \{e, p\}$ and $v \in P$:*

$$x_{v'}^{lb,\min}(\overline{F}) \leq x_v^{lb,\min}(\overline{F}) \quad \text{and} \quad x_v^{ub,\min}(\overline{F}) \leq x_{v'}^{ub,\min}(\overline{F})$$
$$x_{v'}^{lb,\max}(\overline{F}) \leq x_v^{lb,\max}(\overline{F}) \quad \text{and} \quad x_v^{ub,\max}(\overline{F}) \leq x_{v'}^{ub,\max}(\overline{F})$$

where v' is the unique element of P' such that $v \subseteq v'$ and $\overline{F} = \{F\}$.

3.2 Examples

In this section, we present examples illustrating the application of the results given above. Then, in the following section, we provide proofs of Theorems 1 and 2.

Example 2 Let us return to the previous simple example (see Example 1 and Fig. 1). Suppose that we are interested in the minimum probability in the original MDP (Fig. 1(a)) of, starting from state s_0, reaching the target set $\{s_6\}$. This can be computed as 0.15. From the abstraction shown in Fig. 1(c) and the results of Theorem 1, we can establish that the minimum probability lies within the interval [0, 0.5]. If, on the other hand, the abstract model had instead been constructed as an MDP, i.e. with no discrimination between the two forms of nondeterminism, we would only have been able to determine that the minimum reachability probability lay in the interval [0, 1].

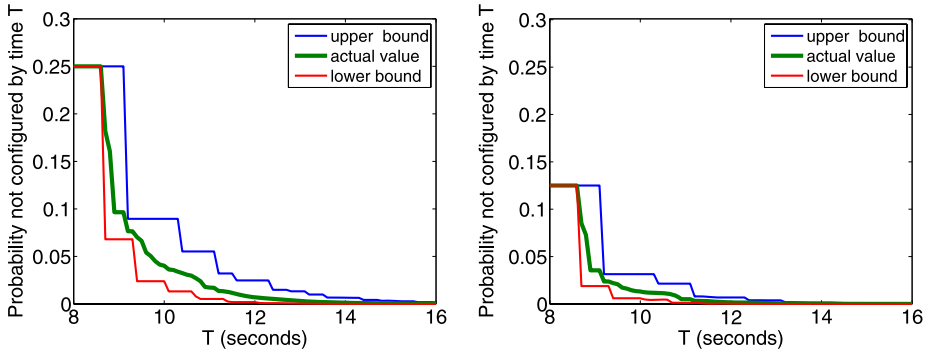
Example 3 To illustrate the application of our abstraction to a larger MDP, we consider a more complex example: a model of the Zeroconf protocol [7] for dynamic self-configuration of local IP addresses within a local network. Zeroconf provides a distributed, 'plug and play' approach to IP address configuration, managed by the individual devices of the network. The model concerns the situation where a new device joins a network of N existing hosts, in which there are a total of M IP addresses available. For full details of the MDP model of the protocol and partition used in the abstraction process, see [28].

Table 1 shows the size of the concrete model (MDP) and its abstraction (game) for a range of values of N and M. For each model, we compute the minimum and maximum expected time for a host to complete the protocol (i.e. to configure and start using an IP address). Table 1 shows the exact results, obtained from the MDP, and the lower and upper bounds, obtained from the abstraction. In addition, Fig. 2 illustrates results for the maximum probability that the new host has not configured successfully by time T.

As stated previously, an advantage of our approach is the ability to quantify the utility of the abstraction, based on the difference between the lower and upper bounds obtained. In the case of the plots in Fig. 2, for a particular time bound T this difference is indicated by the vertical distance between the curves for the lower and upper bounds at the point T on the horizontal axis. Examining these differences between bounds for the presented results,

Table 1 Zeroconf protocol (Example 3): model statistics and numerical results (expected completion time)

N	M	States (transitions)				Minimum expected time			Maximum expected time		
		Concrete model		Abstraction		*lb*	Exact	*ub*	*lb*	Exact	*ub*
4	32	26,121	(50,624)	737	(1,594)	8.1088	8.1572	8.2190	8.1231	8.2465	8.3047
5		58,497	(139,104)	785	(1,678)	8.1410	8.2035	8.2839	8.1595	8.3183	8.3950
6		145,801	(432,944)	833	(1,762)	8.1758	8.2533	8.3538	8.1988	8.3956	8.4922
7		220,513	(614,976)	857	(1,806)	8.2131	8.2939	8.4293	8.2412	8.4579	8.5972
8		432,185	(1,254,480)	881	(1,850)	8.2538	8.3379	8.5110	8.2871	8.5254	8.7109
4	64	50,377	(98,080)	737	(1,594)	8.0508	8.0733	8.1022	8.0574	8.1150	8.1422
5		113,217	(270,272)	785	(1,678)	8.0645	8.0931	8.1299	8.0730	8.1457	8.1808
6		282,185	(839,824)	833	(1,762)	8.0788	8.1136	8.1586	8.0891	8.1774	8.2206
7		426,529	(1,189,792)	857	(1,806)	8.0935	8.1289	8.1883	8.1058	8.2008	8.2619
8		838,905	(2,439,600)	881	(1,850)	8.1088	8.1448	8.2190	8.1231	8.2252	8.3047



(a) N=8 and M=32 (b) N=8 and M=64

Fig. 2 Zeroconf protocol (Example 3): maximum probability the new host has not configured successfully by time T

it can be seen that our abstraction approach yields tight approximations for the performance characteristics of the protocol while at the same time producing a significant reduction in both the number of states and transitions. Observe also that the size of the abstract model increases linearly, rather than exponentially, in N and is independent of M.

3.3 Correctness of the abstraction

We now prove Theorems 1 and 2, presented in Sect. 3.1. Before doing so, we require a number of preliminary concepts. For the remainder of this section we fix an MDP $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$, partitions P and P' of S such that $P \preceq P'$ and set of target states $F \in P \cap P'$. Let $\mathsf{G}^P_{\mathsf{M}} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$ and $\mathsf{G}^{P'}_{\mathsf{M}} = ((V', E'), v'_{init}, (V'_1, V'_2, V_\mathrm{p}'), \delta', \overline{\mathbf{r}}')$ be the stochastic games obtained from P and P' according to Definition 10 and let $\overline{F} = \{F\}$.

To simplify notation, we will add subscripts to distributions $\overline{\mu}$ and sets of reward-distributions pairs $\overline{Steps(s)}$, indicating the partition that was used to lift them from the concrete to the abstract state space, e.g. for partition P, $\mu \in \mathsf{Dist}(S)$, $v \in P$ and $s \in S$, we have $\overline{\mu}_P(v) = \sum_{s \in v} \mu(s)$ and $\overline{Steps(s)}_P = \{(\mathbf{r}(s, \mu), \overline{\mu}_P) \mid \mu \in Steps(s)\}$.

We next define a mapping from vertices of $\mathsf{G}^P_{\mathsf{M}}$ to the vertices of $\mathsf{G}^{P'}_{\mathsf{M}}$.

Definition 12 Let $[\cdot] : V \to V'$ be the mapping from vertices of $\mathsf{G}^P_{\mathsf{M}}$ to vertices of $\mathsf{G}^{P'}_{\mathsf{M}}$ such that for any $v \in V$:

- if $v \in P$, then $[v] = v'$ where v' is the unique element of P' such that $v \subseteq v'$ (such a v' exists from the fact that $P \preceq P'$);
- if $v = \overline{Steps(s)}_P$ for some $s \in S$, then $[v] = \overline{Steps(s)}_{P'}$;
- if $v = (r, \overline{\mu}_P)$ for some $r \in \mathbb{R}_{\geq 0}$ and $\mu \in \mathsf{Dist}(S)$, then $[v] = (r, \overline{\mu}_{P'})$.

By construction, this mapping extends naturally to plays, i.e. for any play $\omega = v_0 v_1 v_2 \ldots$ of $\mathsf{G}^P_{\mathsf{M}}$ we have $[\omega] = [v_0][v_1][v_2] \ldots$ is a play of $\mathsf{G}^{P'}_{\mathsf{M}}$. We denote by $[\cdot]^{-1}$ the inverse mapping from plays of $\mathsf{G}^{P'}_{\mathsf{M}}$ to sets of plays of $\mathsf{G}^P_{\mathsf{M}}$, i.e. for any finite play ω' of $\mathsf{G}^{P'}_{\mathsf{M}}$ we have $[\omega']^{-1} = \{\omega \in V^* \mid [\omega] = \omega'\}$.

Lemma 3 *For any play ω of $\mathsf{G}^P_{\mathsf{M}}$, it holds that $\overline{r}(\overline{F}, \omega) = \overline{r}'(\overline{F}, [\omega])$.*

Proof The proof follows directly from Definition 12 and since we assume that F is an element of both P and P'. □

Using this mapping between plays, for any player 1 vertex v and strategy pair $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ of the game $\mathsf{G}_{\mathsf{M}}^{P}$, we now construct a (randomised) strategy pair $[\boldsymbol{\sigma}]_v = ([\sigma_1]_v, [\sigma_2]_v)$ of the game $\mathsf{G}_{\mathsf{M}}^{P'}$ such that, under $\boldsymbol{\sigma}$ and $[\boldsymbol{\sigma}]_v$, when starting at v and $[v]$ respectively, the probability of reaching F is equal. For any finite play ω' of $\mathsf{G}_{\mathsf{M}}^{P'}$ such that $\omega'(0) = [v]$ and $last(\omega') \in V'_i$, let the probability of $[\sigma_i]_v$ selecting vertex $v' \in V'$ after play ω' equal:

$$[\sigma_i]_v(\omega')(v') \stackrel{\text{def}}{=} \frac{Prob_v^{\boldsymbol{\sigma}}([\omega' v']_v^{-1})}{Prob_v^{\boldsymbol{\sigma}}([\omega']_v^{-1})}$$

where $[\tilde{\omega}']_v^{-1} = \{\omega \in [\tilde{\omega}']^{-1} \mid \omega(0) = v\}$ for any play $\tilde{\omega}'$ of $\mathsf{G}_{\mathsf{M}}^{P'}$ and

$$Prob_v^{\boldsymbol{\sigma}}(\Omega) = Prob_v^{\boldsymbol{\sigma}}\{\omega \in Play^{\vec{\sigma}}(v) \mid \omega^{(i)} \in \Omega \text{ for some } i \in \mathbb{N}\}$$

for any set of finite plays Ω of $\mathsf{G}_{\mathsf{M}}^{P}$.

Proposition 1 *For any vertex v and strategy pair $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ of $\mathsf{G}_{\mathsf{M}}^{P}$, if Ω is a measurable set of plays of $Play^{[\vec{\sigma}]_v}([v])$, then $Prob_{[v]}^{[\boldsymbol{\sigma}]_v}(\Omega) = Prob_v^{\boldsymbol{\sigma}}([\Omega]_v^{-1})$.*

Proof From the measure construction (see Sect. 2) it is sufficient to show that:

$$Prob_{[v]}^{[\boldsymbol{\sigma}]_v}(\omega') = Prob_v^{\boldsymbol{\sigma}}[\omega']_v^{-1}$$

for any finite play ω' of $\mathsf{G}_{\mathsf{M}}^{P'}$ and vertex $v \in [\omega'(0)]^{-1}$, which we now prove by induction on the length of the play ω'. If $|\omega'| = 1$, then ω' comprises a single vertex and since $[\omega']_v^{-1} = \{v\}$, by the measure construction:

$$Prob_{[v]}^{[\boldsymbol{\sigma}]_v}(\omega') = Prob_v^{\boldsymbol{\sigma}}[\omega']_v^{-1} = 1.$$

Next, suppose by induction that the lemma holds for all plays of length n. Consider any play ω' of length $n+1$ of $\mathsf{G}_{\mathsf{M}}^{P'}$. By definition ω' is of the form $\tilde{\omega}'\tilde{v}'$ for some play $\tilde{\omega}'$ of length n and vertex $\tilde{v}' \in V'$. If $last(\tilde{\omega}') \in V'_i$ (for $i = 1, 2$), then from the measure construction:

$$\begin{aligned}
Prob_{[v]}^{[\boldsymbol{\sigma}]_v}(\omega') &= Prob_{[v]}^{[\boldsymbol{\sigma}]_v}(\tilde{\omega}') \cdot [\sigma_i]_v(\tilde{\omega}')(\tilde{v}') \\
&= Prob_v^{\boldsymbol{\sigma}}([\tilde{\omega}']_v^{-1}) \cdot [\sigma_i]_v(\tilde{\omega}')(\tilde{v}') && \text{by induction} \\
&= Prob_v^{\boldsymbol{\sigma}}([\tilde{\omega}']_v^{-1}) \cdot \frac{Prob_v^{\boldsymbol{\sigma}}([\tilde{\omega}'\tilde{v}']_v^{-1})}{Prob_v^{\boldsymbol{\sigma}}([\tilde{\omega}']_v^{-1})} && \text{by definition of } [\sigma_i]_v \\
&= Prob_v^{\boldsymbol{\sigma}}([\tilde{\omega}\tilde{v}']_v^{-1}) && \text{rearranging} \\
&= Prob_v^{\boldsymbol{\sigma}}([\omega']_v^{-1}) && \text{since } \omega' = \tilde{\omega}'\tilde{v}'.
\end{aligned}$$

On the other hand, if $last(\tilde{\omega}') \in V'_{\mathrm{p}}$, then $last(\tilde{\omega}') = (r, \overline{\mu_{P'}})$ for some $r \in \mathbb{R}_{\geq 0}$ and $\mu \in \mathsf{Dist}(S)$, and hence in this case, again from the measure construction, we have:

$$
\begin{aligned}
Prob^{[\boldsymbol{\sigma}]_v}_{[v]}(\omega') &= Prob^{[\boldsymbol{\sigma}]_v}_{[v]}(\tilde{\omega}') \cdot \overline{\mu_{P'}}(\tilde{v}') \\
&= Prob^{\boldsymbol{\sigma}}_{v}([\tilde{\omega}']^{-1}_{v}) \cdot \overline{\mu_{P'}}(\tilde{v}') && \text{by induction} \\
&= Prob^{\boldsymbol{\sigma}}_{v}([\tilde{\omega}']^{-1}_{v}) \cdot \left(\sum_{\tilde{v} \in [\tilde{v}']^{-1}} \overline{\mu_P}(\tilde{v}) \right) && \text{by Definition 12} \\
&= \sum_{\tilde{v} \in [\tilde{v}']^{-1}} Prob^{\boldsymbol{\sigma}}_{v}([\tilde{\omega}']^{-1}_{v}) \cdot \overline{\mu_P}(\tilde{v}) && \text{rearranging} \\
&= \sum_{\tilde{v} \in [\tilde{v}']^{-1}} Prob^{\boldsymbol{\sigma}}_{v}([\tilde{\omega}']^{-1}_{v} \tilde{v}) && \text{by definition of } Prob^{\boldsymbol{\sigma}}_{v} \\
&= Prob^{\boldsymbol{\sigma}}_{v}([\omega']^{-1}_{v}) && \text{by Definition 12 and since } \omega' = \tilde{\omega}'\tilde{v}'.
\end{aligned}
$$

Since these are all the cases to consider, the lemma holds by induction. □

Before we give the proofs of Theorems 1 and 2 we require the following lemma, a classical result from measure theory and two propositions.

Lemma 4 *For any vertex v and strategy pair $\boldsymbol{\sigma} = (\sigma_1, \sigma_2)$ of $\mathsf{G}^P_{\mathsf{M}}$, the mapping $[\cdot]$ of Definition 12 is a measurable function between $(Play^{\boldsymbol{\sigma}}(v), \mathcal{F}^{\boldsymbol{\sigma}}_v)$ and $(Play^{[\boldsymbol{\sigma}]_v}([v]), \mathcal{F}^{[\boldsymbol{\sigma}]_v}_{[v]})$.*

Proof The proof follows from measure construction (see Sect. 2). □

Theorem 3 [3] *Let $(\Omega, \mathcal{F})$ and $(\Omega', \mathcal{F}')$ be measurable spaces, and suppose that Pr is a measure on $(\Omega, \mathcal{F})$ and the function $T : \Omega \to \Omega'$ is measurable. If f is a real non-negative measurable function on $(\Omega', \mathcal{F}')$, then*:

$$\int_{\omega \in \Omega} f(T\omega)\, \mathrm{d}\, Pr = \int_{\omega' \in \Omega'} f(\omega')\, \mathrm{d}\, Pr\, T^{-1}.$$

Proposition 2 *For any vertex v and strategy pair $\boldsymbol{\sigma}$ of $\mathsf{G}^P_{\mathsf{M}}$ we have $p^{\boldsymbol{\sigma}}_v(\overline{F}) = p^{[\vec{\sigma}]_v}_{[v]}(\overline{F})$ and $e^{\boldsymbol{\sigma}}_v(\overline{F}) = e^{[\vec{\sigma}]_v}_{[v]}(\overline{F})$.*

Proof The first equation follows from Proposition 1 using standard results from measure theory. For the second equation, by definition of the expected rewards for games (see Sect. 2.4), we have:

$$
\begin{aligned}
e^{\boldsymbol{\sigma}}_v(\overline{F}) &= \int_{\omega \in Play^{\boldsymbol{\sigma}}(v)} \overline{\mathbf{r}}(\overline{F}, \omega)\, \mathrm{d} Prob^{\boldsymbol{\sigma}}_v \\
&= \int_{\omega \in Play^{\boldsymbol{\sigma}}(v)} \overline{\mathbf{r}}'(\overline{F}, [\omega])\, \mathrm{d} Prob^{\boldsymbol{\sigma}}_v && \text{by Lemma 3} \\
&= \int_{\omega' \in Play^{[\boldsymbol{\sigma}]_v}([v])} \overline{\mathbf{r}}(\overline{F}, \omega')\, \mathrm{d} Prob^{\boldsymbol{\sigma}}_v([\cdot]^{-1}) && \text{by Theorem 3 and Lemma 4} \\
&= \int_{\omega' \in Play^{[\boldsymbol{\sigma}]_v}([v])} \overline{\mathbf{r}}(\overline{F}, \omega')\, \mathrm{d} Prob^{[\boldsymbol{\sigma}]_v}_{[v]} && \text{by Proposition 1} \\
&= e^{[\boldsymbol{\sigma}]_v}_{[v]}(\overline{F})
\end{aligned}
$$

as required. □

Proposition 3 *For any vertex v' and player 2 strategy σ_2' of $\mathsf{G}_{\mathsf{M}}^{P'}$ there exists a player 2 strategy σ_2 of $\mathsf{G}_{\mathsf{M}}^{P}$ such that*

$$\sup_{\sigma_1'} x_{v'}^{\sigma_1',\sigma_2'}(\overline{F}) \geq \sup_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$\inf_{\sigma_1'} x_{v'}^{\sigma_1',\sigma_2'}(\overline{F}) \leq \inf_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

for any vertex v of $\mathsf{G}_{\mathsf{M}}^{P}$ such that $[v] = v'$ and $x \in \{e, p\}$.

Proof Let v' be a vertex and σ_2' a player 2 strategy of $\mathsf{G}_{\mathsf{M}}^{P'}$ and let v be any vertex of $\mathsf{G}_{\mathsf{M}}^{P}$ such that $[v] = v'$.

In the first step of the proof we construct the player 2 strategy σ_2. Therefore consider any finite play ω of $\mathsf{G}_{\mathsf{M}}^{P}$ that ends in a player 2 vertex. By Definition 10 we have $last(\omega) = \overline{Steps(s)_P}$ for some $s \in S$ and $[\omega]$ is a finite play of $\mathsf{G}_{\mathsf{M}}^{P'}$ ending in the player 2 vertex $\overline{Steps(s)_{P'}}$. Now $\sigma_2'([\omega]) = (\mathbf{r}(s,\mu), \overline{\mu_{P'}})$ for some $\mu \in Steps(s)$ and we let $\sigma_2(\omega) = (\mathbf{r}(s,\mu), \overline{\mu_P})$ which by Definition 10 is an element of $\overline{Steps(s)_P}$, and hence is a valid player 2 choice in $\mathsf{G}_{\mathsf{M}}^{P}$. Although the μ meeting the above requirements is not necessarily unique, for the purposes of the proof, it is sufficient to choose any such distribution. From Definition 12 it then follows that:

$$[\omega(\sigma_2(\omega))] = [\omega]\sigma_2'([\omega]) \quad \text{for all finite plays } \omega \text{ of } \mathsf{G}_{\mathsf{M}}^{P} \text{ that end in a player 2 vertex.} \tag{2}$$

In the next step of the proof we will show that, for any player 1 strategy σ_1 of $\mathsf{G}_{\mathsf{M}}^{P}$, if $([\sigma_1]_v, [\sigma_2]_v)$ is the strategy pair of $\mathsf{G}_{\mathsf{M}}^{P'}$ constructed from the strategy pair (σ_1, σ_2) of $\mathsf{G}_{\mathsf{M}}^{P}$ following the method described above, then $[\sigma_2]_v(\omega) = \sigma_2'(\omega)$ for all finite plays ω' of $\mathsf{G}_{\mathsf{M}}^{P'}$ such that $\omega'(0) = [v]$ and $last(\omega')$ is a player 2 vertex. Therefore, consider any such play ω' of $\mathsf{G}_{\mathsf{M}}^{P'}$. Now, by construction of $([\sigma_1]_v, [\sigma_2]_v)$ we have that for any $v' \in V'$:

$$[\sigma_2]_v(\omega')(v') = \frac{Prob_v^{\sigma}([\omega' v']_v^{-1})}{Prob_v^{\sigma}([\omega']_v^{-1})}$$

where $[\tilde{\omega}']_v^{-1} = \{\omega \in [\tilde{\omega}']^{-1} \mid \omega(0) = v\}$ for any play $\tilde{\omega}'$ of $\mathsf{G}_{\mathsf{M}}^{P'}$ and

$$Prob_v^{\sigma_1,\sigma_2}(\Omega) = Prob_v^{\sigma_1,\sigma_2}\{\omega \in Play^{\sigma_1,\sigma_2}(v) \mid \omega^{(i)} \in \Omega \text{ for some } i \in \mathbb{N}\}$$

for any set of finite plays Ω of $\mathsf{G}_{\mathsf{M}}^{P}$. Now, from (2) we have that:

$$\{\omega \in Play^{\sigma_1,\sigma_2}(v) \mid [\omega] = \omega' v'\} = \begin{cases} \{\omega(\sigma_2(\omega)) \in Play^{\sigma_1,\sigma_2}(v) \mid [\omega] = \omega'\} & \text{if } v' = \sigma_2'([\omega]) \\ \emptyset & \text{otherwise.} \end{cases}$$

Combining this with the definition of $[\sigma_2]_v$, it follows that:

$$[\sigma_2]_v(\omega')(v') = \begin{cases} 1 & \text{if } v' = \sigma_2'([\omega]) \\ 0 & \text{otherwise,} \end{cases}$$

and hence, since ω' was arbitrary, we have that $[\sigma_2]_v(\omega) = \sigma_2'(\omega)$ for all finite play ω' of $\mathsf{G}_{\mathsf{M}}^{P'}$ such that $\omega'(0) = [v]$ and $last(\omega')$ is a player 2 vertex.

In the final step of the proof, we will use these results to show that the proposition holds. Since v was arbitrary, using Proposition 2 and the above correspondence between σ_2 and $[\sigma_2]$, we have that for any player 1 strategy σ_1 of $\mathsf{G}_{\mathsf{M}}^P$:

$$x_{v'}^{[\sigma_1],\sigma_2'}(\overline{F}) = x_v^{\sigma_1,\sigma_2}(\overline{F})$$

for all vertices v of $\mathsf{G}_{\mathsf{M}}^P$ such that $[v] = v'$ and $x \in \{e, p\}$. Hence, since σ_1 was arbitrary, it follows that:

$$\sup_{\sigma_1'} x_{v'}^{\sigma_1',\sigma_2'}(\overline{F}) \geq \sup_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$\inf_{\sigma_1'} x_{v'}^{\sigma_1',\sigma_2'}(\overline{F}) \leq \inf_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

for any vertex v of $\mathsf{G}_{\mathsf{M}}^P$ such that $[v] = v'$ and $x \in \{e, p\}$ as required. □

We are now in a position to give the proofs of Theorems 1 and 2.

Proof of Theorem 2 Consider any $x \in \{e, p\}$ and $v \in P$. From Proposition 2 it follows that:

$$\inf_{\sigma_1',\sigma_2'} x_{[v]}^{\sigma_1',\sigma_2'}(\overline{F}) \leq \inf_{\sigma_1,\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

$$\sup_{\sigma_1',\sigma_2'} x_{[v]}^{\sigma_1',\sigma_2'}(\overline{F}) \geq \sup_{\sigma_1,\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F})$$

and hence $x_{[v]}^{lb,\min}(\overline{F}) \leq x_v^{lb,\min}(\overline{F})$ and $x_v^{ub,\max}(\overline{F}) \leq x_{[v]}^{ub,\max}(\overline{F})$.

On the other hand, by definition we have:

$$\begin{aligned}
x_v^{ub,\min}(\overline{F}) &= \sup_{\sigma_1'} \inf_{\sigma_2'} x_{[v]}^{\sigma_1',\sigma_2'}(\overline{F}) \\
&= \inf_{\sigma_2'} \sup_{\sigma_1'} x_{[v]}^{\sigma_1',\sigma_2'}(\overline{F}) && \text{by [9, 40]} \\
&\geq \inf_{\sigma_2} \sup_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F}) && \text{by Proposition 3} \\
&= \sup_{\sigma_1} \inf_{\sigma_2} x_v^{\sigma_1,\sigma_2}(\overline{F}) && \text{by [9, 40]} \\
&= x_{[v]}^{ub,\min}(\overline{F}) && \text{by definition.}
\end{aligned}$$

Similarly, by definition and [9, 40]:

$$\begin{aligned}
x_{[v]}^{lb,\max}(\overline{F}) &= \sup_{\sigma_2'} \inf_{\sigma_1'} x_{[v]}^{\sigma_1',\sigma_2'}(\overline{F}) \\
&\geq \sup_{\sigma_2} \inf_{\sigma_1} x_v^{\sigma_1,\sigma_2}(\overline{F}) && \text{by Proposition 3} \\
&= x_v^{lb,\max}(\overline{F}) && \text{by definition and [9, 40]}
\end{aligned}$$

which completes the proof. □

Proof of Theorem 1 The proof follows from Theorem 2 together with the following facts.

– The partition $P = \{\{s\} \mid s \in S\}$ is finer than all other partitions.
– If $\mathsf{G}^P_{\mathsf{M}}$ is constructed from partition $P = \{\{s\} \mid s \in S\}$ according to Definition 10, then:

$$x^{lb,\min}_{\{s\}}(\overline{F}) = x^{\min}_s(F) = x^{ub,\min}_{\{s\}}(\overline{F})$$
$$x^{lb,\max}_{\{s\}}(\overline{F}) = x^{\max}_s(F) = x^{ub,\max}_{\{s\}}(\overline{F})$$

for all $s \in S$ and $x \in \{e, p\}$.

□

4 An abstraction-refinement framework for Markov decision processes

We now present a framework for automatically generating abstractions of MDPs, which uses the game-based abstraction method of Sect. 3. This is inspired by the success of counterexample-guided abstraction-refinement (CEGAR) techniques, developed for non-probabilistic verification, which iteratively *refine* an abstraction based on results from model checking. Probabilistic verification, however, is typically *quantitative* in nature: probabilistic model checking of MDPs essentially involves computing minimum or maximum probabilistic or expected reachability measures. Thus, we propose a *quantitative* abstraction-refinement process.

The construction and analysis of a game-based abstraction of an MDP yields lower and upper bounds for these values (plus corresponding strategies). The difference between these bounds can be seen as a measure of the quality of the abstraction. If this difference is too high (say, above some error ε), then it is desirable to refine the abstraction to reduce the difference. This provides the basis for a quantitative abstraction-refinement loop, illustrated in Fig. 3: starting with a simple, coarse abstraction, we refine the abstraction until the difference between the bounds is below ε. In fact we can consider different criteria for termination of the process: for example checking that the difference is below ε for a single abstract state or set of abstract states, such as the abstract states containing an initial state.

4.1 Refinement strategies

In our context, refinement corresponds to replacing the partition used to build the abstraction with a finer partition. We propose two methods, called *strategy-based* and *value-based* refinement. The first examines the difference between strategy pairs that yield the lower and upper bounds. The motivation for this is that, since the bounds are different and the actual

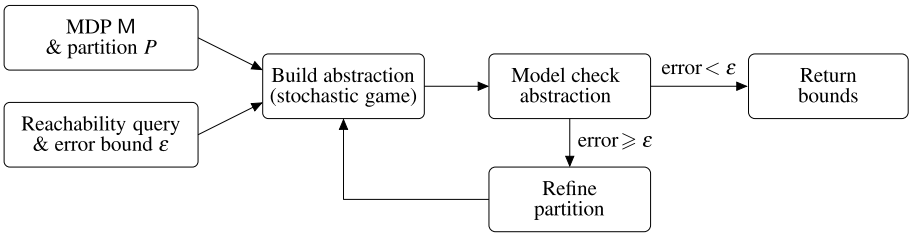


Fig. 3 A quantitative abstraction-refinement framework for MDPs

value for the concrete model falls between the bounds, one (or both) of the strategy pairs must make choices that are not valid in the concrete model (this can be seen as analogous to non-probabilistic abstraction refinement, which is based on a counterexample trace). The second method differs by considering all strategy pairs yielding the bounds.

For both methods, we demonstrate that the refinement results in a strictly finer partition. For finite-state models, this guarantees that, for any user-specified error bound ε, the abstraction-refinement loop eventually terminates. In addition, if a partition is coarser than *probabilistic bisimulation* [31], i.e. bisimilar states are in the same element of the partition, then so is the partition after refinement. This follows from the fact that, under such a partition, bisimilar states are represented by the same player 2 vertex and, as will be seen, neither refinement method separates such states. Consequently, if the initial partition is coarser than probabilistic bisimulation, then any subsequent refinement will also be coarser. This observation guarantees that the refinement process also terminates when it is applied to infinite-state MDPs that have a finite bisimulation quotient.

To simplify presentation, we describe the refinement step when computing *minimum* reachability probabilities, i.e. we assume that we have MDP $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$, partition P, target set $F \in P$ and corresponding game $\mathsf{G}^P_{\mathsf{M}} = ((V, E), v_{init}, (V_1, V_2, V_{\mathrm{p}}), \delta, \overline{\mathbf{r}})$ in which $p_v^{lb,\min}(\overline{F}) < p_v^{ub,\min}(\overline{F})$ for some $v \in V_1$. The cases for maximum probabilistic and expected reachability follow in identical fashion.

Strategy-based refinement As mentioned in Sect. 2, when computing $p_v^{lb,\min}(\overline{F})$ and $p_v^{ub,\min}(\overline{F})$, a pair of strategies that obtain these values is also generated. Since a player 1 strategy's choice can be considered as a set of concrete states (the states whose choices are abstracted to the chosen player 2 vertex), the player 1 strategy from each pair provides a method for refinement. More precisely, we refine P as follows.

1. Find memoryless strategy pairs $(\sigma_1^{lb}, \sigma_2^{lb})$, $(\sigma_1^{ub}, \sigma_2^{ub})$ such that for any $v \in V$:

$$p_v^{\sigma_1^{lb},\sigma_2^{lb}}(\overline{F}) = p_v^{lb,\min}(\overline{F}) \quad \text{and} \quad p_v^{\sigma_1^{ub},\sigma_2^{ub}}(\overline{F}) = p_v^{ub,\min}(\overline{F}).$$

2. For each player 1 vertex $v \in V_1$ such that $\sigma_1^{lb}(v) \neq \sigma_1^{ub}(v)$ replace v in the partition P with $v_{lb}^{\sigma_1}$, $v_{ub}^{\sigma_1}$ and $v \setminus (v_{lb}^{\sigma_1} \cup v_{ub}^{\sigma_1})$ where:

$$v_{lb}^{\sigma_1} = \left\{ s \in v \mid \sigma_1^{lb}(v) = \overline{Steps(s)} \right\} \quad \text{and} \quad v_{ub}^{\sigma_1} = \left\{ s \in v \mid \sigma_1^{ub}(v) = \overline{Steps(s)} \right\}.$$

The following states that there always exists a vertex v such that $\sigma_1^{lb}(v) \neq \sigma_1^{ub}(v)$, and hence applying this refinement will always (strictly) refine the partition.

Lemma 5 *If there exists a vertex $v' \in V_1$ such that $p_{v'}^{lb,\min}(\overline{F}) < p_{v'}^{ub,\min}(\overline{F})$ and $(\sigma_1^{lb}, \sigma_2^{lb})$ and $(\sigma_1^{ub}, \sigma_2^{ub})$ are corresponding memoryless optimal strategy pairs, then there also exists a vertex $v \in V_1$ such that $\sigma_1^{lb}(v) \neq \sigma_1^{ub}(v)$.*

Value-based refinement The second refinement method is again based on the choices made by optimal strategy pairs. However, rather than looking at a single strategy pair for each of the bounds, we use all strategies that achieve one of the bounds when refining the partition. More precisely, letting $v_2^s = \overline{Steps(s)}$, we refine each element $v \in P$ as follows.

1. Construct the following subsets of v:

$$v_{lb}^{\min} = \{s \in v \mid p_{v_2^s}^{lb,\min}(\overline{F}) = p_v^{lb,\min}(\overline{F})\}$$

$$v_{ub}^{\min} = \{s \in v \mid p_{v_2(s)}^{ub,\min}(\overline{F}) = p_v^{ub,\min}(\overline{F})\}.$$

2. Replace v with $v_{lb}^{\min} \setminus v_{ub}^{\min}$, $v_{ub}^{\min} \setminus v_{lb}^{\min}$, $v_{lb}^{\min} \cap v_{ub}^{\min}$ and $v \setminus (v_{lb}^{\min} \cup v_{ub}^{\min})$.

The following states that this refinement method will always lead to a (strictly) finer partition. Recall that the family of pure memoryless strategies suffices for optimality with respect to probabilistic and expected reachability in stochastic games.

Lemma 6 *If there exists $v' \in V_1$ such that $p_{v'}^{lb,\min}(\overline{F}) < p_{v'}^{ub,\min}(\overline{F})$, then there exists $v \in V_1$ such that $v_{lb}^{\min} \neq v_{ub}^{\min}$.*

Proof The proof is by contradiction. Therefore, suppose that there exists $v' \in V_1$ such that $p_{v'}^{lb,\min}(\overline{F}) < p_{v'}^{ub,\min}(\overline{F})$ and $v_{lb}^{\min} = v_{ub}^{\min}$ for all $v \in V_1$. Now, consider any memoryless strategy pairs $(\sigma_1^{lb}, \sigma_2^{lb})$ and $(\sigma_1^{ub}, \sigma_2^{ub})$ such that:

$$p_v^{\sigma_1^{lb},\sigma_2^{lb}}(\overline{F}) = p_v^{lb,\min}(\overline{F}) \quad \text{and} \quad p_v^{\sigma_1^{ub},\sigma_2^{ub}}(\overline{F}) = p_v^{ub,\min}(\overline{F}) \quad \text{for all } v \in V$$

whose existence follows from [9, 40]. Now, for any player 1 vertex $v \in V_1$, by definition:

$$v_{lb}^{\min} = \{s \in v \mid p_{v_2(s)}^{lb,\min}(\overline{F}) = p_v^{lb,\min}(\overline{F})\}$$

$$v_{ub}^{\min} = \{s \in v \mid p_{v_2(s)}^{ub,\min}(\overline{F}) = p_v^{ub,\min}(\overline{F})\}$$

where $v_2(s) = \overline{Steps(s)}$. Now, consider any $v \in V_1$, by construction we have $\sigma_1^{ub}(v) = v_2^{s_{ub}}$ for some $s_{ub} \in v_{ub}^{\min}$, and since by the hypothesis $v_{lb}^{\min} = v_{ub}^{\min}$, we also have that $s_{ub} \in v_{lb}^{\min}$. Therefore, since σ_2^{lb} is optimal and $v \in V_1$ was arbitrary, it follows that:

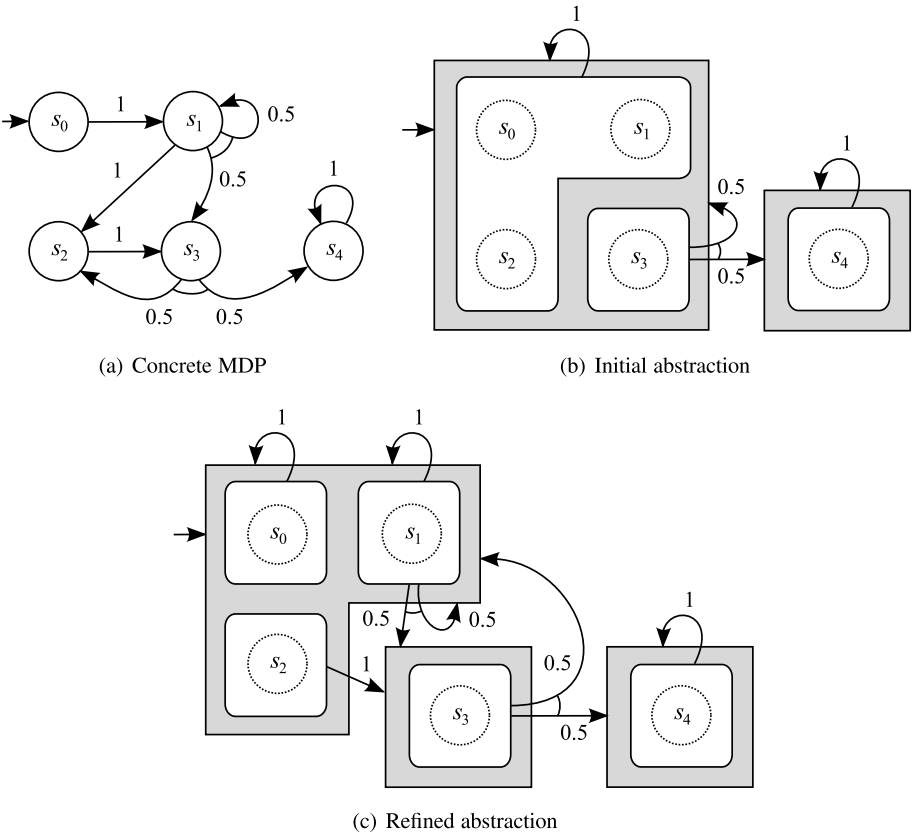
$$p_{v'}^{lb,\min}(\overline{F}) = p_{v'}^{\sigma_1^{ub},\sigma_2^{lb}}(\overline{F}) \quad \text{for all } v \in V_1.$$

In particular, since v' is a player 1 vertex, we have:

$$\begin{aligned}
p_{v'}^{lb,\min}(\overline{F}) &= p_{v'}^{\sigma_1^{ub},\sigma_2^{lb}}(\overline{F}) \\
&\geq \inf_{\sigma_2} p_{v'}^{\sigma_1^{ub},\sigma_2}(\overline{F}) && \text{since } \sigma_2^{lb} \text{ is a player 2 strategy} \\
&= \sup_{\sigma_1} \inf_{\sigma_2} p_{v'}^{\sigma_1,\sigma_2}(\overline{F}) && \text{since } \sigma_1^{ub} \text{ is an optimal strategy} \\
&= p_{v'}^{ub,\min}(\overline{F}) && \text{by definition}
\end{aligned}$$

which contradicts the fact that $p_{v'}^{lb,\min}(\overline{F}) < p_{v'}^{ub,\min}(\overline{F})$. □

Example 4 To illustrate the refinement strategies, we consider the simple MDP given in Fig. 4(a). We assume that a reward of 1 is associated with each transition and compute the maximum expected reward of reaching the target set $\{s_4\}$. We start by partitioning the state space into the target set ($\{s_4\}$) and the remaining states, resulting in the stochastic two-player



(a) Concrete MDP (b) Initial abstraction

(c) Refined abstraction

Fig. 4 Example of the abstraction-refinement process

game shown in Fig. 4(b). As in Example 1, player 1 vertices are drawn as grey rectangles, each one's successor player 2 vertices are drawn as white shapes enclosed within it, and probabilistic vertices are represented by sets of grouped, labelled arrows. The states of the original MDP, sets of which comprise player 1 and player 2 vertices, are also depicted.

Calculating the lower and upper bounds on the maximum expected reward to reach $\{s_4\}$, using the game in Fig. 4(b), yields the interval $[2, \infty)$ for the vertex containing states s_0–s_3 (and $[0, 0]$ for the one containing s_4). There is only one memoryless strategy resulting in each bound: the one that selects $\{s_3\}$ (for the lower bound) and the one that selects $\{s_0, s_1, s_2\}$ (for the upper bound). Consequently, both the strategy-based and value-based refinement techniques refine the partition from $\{s_0, s_1, s_2, s_3\}, \{s_4\}$ to $\{s_0, s_1, s_2\}, \{s_3\}, \{s_4\}$. The new abstraction is shown in Fig. 4(c). Note that the player 1 vertex containing $\{s_0, s_1, s_2\}$ now leads to three distinct player 2 vertices, since the (lifted) distributions for MDP states s_0, s_1 and s_2 are no longer equivalent under the new partition. Calculating bounds with the new game now gives $[4, \infty)$ and $[3, \infty)$ for the vertices $\{s_0, s_1, s_2\}$ and $\{s_3\}$. From the vertex $\{s_0, s_1, s_2\}$, the optimal strategy for the lower bound chooses $\{s_2\}$ and for the upper bound can choose either $\{s_0\}$ or $\{s_1\}$. With strategy-based refinement, using either strategy for the upper bound gives the partition $\{s_0\}, \{s_1\}, \{s_2\}, \{s_3\}, \{s_4\}$. For value-based refinement, s_0 and s_1 are kept together, giving rise instead to the partition $\{s_0, s_1\}, \{s_2\}, \{s_3\}, \{s_4\}$.

4.2 The abstraction-refinement algorithm

With the refinement strategies of the previous section, we can implement the abstraction-refinement loop shown in Fig. 3. Starting with a coarse partition, we iteratively: construct an abstraction (according to Definition 10); compute lower and upper bounds (and corresponding strategies); and refine using either the strategy-based or value-based method of Sect. 4.1.

In this section, we present an optimised algorithm for the refinement loop, illustrated in Fig. 5(a). At the ith refinement step, we construct the abstraction $\mathsf{G}_{\mathsf{M}}^{P_i}$ of the MDP using partition P_i. Using $\mathsf{G}_{\mathsf{M}}^{P_i}$, we then compute vectors of lower and upper bounds $\mathbf{x}^{lb,(i)}$ and $\mathbf{x}^{ub,(i)}$ (indexed over P_i, i.e. over player 1 vertices of $\mathsf{G}_{\mathsf{M}}^{P_i}$) for the minimum probability of reaching F in the MDP, using value iteration. If the bounds are sufficiently close (within ε), the loop terminates; otherwise, we refine the partition. The algorithm includes two optimisations, designed to improve the convergence rate of numerical solution:

– wherever possible, we re-use existing numerical results: when calculating lower bounds (line 5), we use the same vector from the previous step as the initial solution; for upper bounds (line 6), we re-use the lower bound from the current step; both are guaranteed lower bounds on the solution.
– we store (in set V_{done}) vertices for which lower and upper bounds coincide (and we thus have the exact solution for their concrete states); again, we use these values in initial solutions for value iteration.

// Initialise partition and probability vector
1. $P_0{:=}P$; $V_{done}{:=}\emptyset$
2. **for** $(v \in P_0)$ $x_v^{lb,(0)}{:=}0$

// Main loop
3. **for** $(i = 0, 1, 2, \ldots)$
4. $\mathsf{G}_{\mathsf{M}}^{P_i}{:=}\textsc{ConstructAbstractMdp}(\mathsf{M}, P_i)$
5. $(\mathbf{x}^{lb,(i)}, \sigma_i^{lb}){:=}\textsc{ValIter}(\mathsf{G}_{\mathsf{M}}^{P_i}, F, lb, P_i \backslash V_{done}, \mathbf{x}^{lb,(i)})$
6. $(\mathbf{x}^{ub,(i)}, \sigma_i^{ub}){:=}\textsc{ValIter}(\mathsf{G}_{\mathsf{M}}^{P_i}, F, ub, P_i \backslash V_{done}, \mathbf{x}^{lb,(i)})$

// Terminate loop if done
7. **if** $(\|\mathbf{x}^{ub,(i)} - \mathbf{x}^{lb,(i)}\| < \varepsilon)$ **return** $(\mathbf{x}^{lb,(i)}, \mathbf{x}^{ub,(i)})$

// Otherwise, refine, build partition P_{i+1}
// and vector $\mathbf{x}^{lb,(i+1)}$ for next iteration
8. $P_{i+1}{:=}\emptyset$
9. **for** $(v \in P_i)$

// Vertices that cannot be refined
10. **if** $(x_v^{lb,(i)} = x_v^{ub,(i)})$
11. $P_{i+1}{:=}P_{i+1} \cup \{v\}$
12. $V_{done}{:=}V_{done} \cup \{v\}$
13. $x_v^{lb,(i+1)}{:=}x_v^{lb}$

// Vertices that can be refined
15. **else**
16. $P'{:=}\textsc{Refine}(\mathsf{G}_{\mathsf{M}}^{P_i}, v, (\mathbf{x}^{lb,(i)}, \sigma_i^{lb}), (\mathbf{x}^{ub,(i)}, \sigma^{ub,(i)}))$
17. $P_{i+1}{:=}P_{i+1} \cup P'$
18. **for** $(v' \in P')$ $x_{v'}^{lb,(i+1)}{:=}x_v^{lb,(i)}$
19. **end**
20. **end**
21. **end**

(a) $\textsc{AbstractRefineLoop}(\mathsf{M}, P, F)$

// Initialise probability vector
1. $\mathbf{x}{:=}\mathbf{x}^{init}$;

// Qualitative pre-computation
2. $V_0^\star{:=}\textsc{Prob0}(\mathsf{G}, F, \star)$
3. **for** $(v_1 \in V_0^\star)$ $x_{v_1}{:=}0$
4. $V_1^\star{:=}\textsc{Prob1}(\mathsf{G}, F, \star)$
5. **for** $(v_1 \in V_1^\star)$ $x_{v_1}{:=}1$

// Main fixpoint loop
6. **repeat**
7. $\hat{\mathbf{x}}{:=}\mathbf{x}$;
8. **if** $(\star = lb)$
9. **for** $(v_1 \in V_? \backslash (V_0^\star \cup V_1^\star))$
10. $x_{v_1} := \min_{v_2 \leftarrow v_1} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot \hat{x}_v$
11. $\sigma(v_1) := \operatorname{argmin}_{v_2 \leftarrow v_1} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot \hat{x}_v$
12. **end**
13. **else**
14. **for** $(v_1 \in V_? \backslash (V_0^\star \cup V_1^\star))$
15. $x_{v_1} := \max_{v_2 \leftarrow v_1} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot \hat{x}_v$
16. $\sigma(v_1) := \operatorname{argmax}_{v_2 \leftarrow v_1} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot \hat{x}_v$
17. **end**
18. **end**
19. **until** $\|\mathbf{x} - \hat{\mathbf{x}}\| < \delta$

// Return vector of probabilities
// and corresponding player 1 strategy
20. **return** $(\mathbf{x}, \sigma)$

(b) $\textsc{ValIter}(\mathsf{G}, F, \star, V_?, \mathbf{x}^{init})$

Fig. 5 Abstraction-refinement loop algorithm (minimum reachability probabilities)

Numerical solution of the game at each step is performed with an improved version of the standard value iteration algorithm [10], illustrated in Fig. 5(b), which includes several optimisations. Firstly, we employ the qualitative algorithms of [16] to efficiently find vertices for which the solution is exactly 0 or 1 (lines 2 and 4). Secondly, we only compute new values for vertices where the answer is unknown, i.e. those not in the set V_{done}, mentioned above, and not identified by the qualitative algorithms (lines 9 and 14). Thirdly, the value iteration algorithm is given an initial vector $\mathbf{x}^{init}$ for the fixpoint computation (normally this is a vector of zeros). This is used, as described above, to improve convergence. The correctness of the value iteration scheme is given in Sect. 4.3 below.

As in the previous section, the code presented in Fig. 5 is for computation of minimum probabilities. The case for maximum probabilities and for expected rewards require only trivial changes to the value iteration algorithm. The REFINE function referenced in line 16 of Fig. 5(a) can be either the strategy-based or value-based refinement from Sect. 4.1. For the latter, we require that lines 11 and 16 of the VALITER algorithm in Fig. 5(b) return all (rather than just one) player 1 strategy choice which achieves the lower/upper bound.

4.3 Correctness of the abstraction-refinement algorithm

We now demonstrate that the value iteration scheme of Fig. 5 converges. For simplicity, we only consider computing bounds on minimum reachability probabilities and assume a fixed MDP $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$ and target set F. The cases for maximum probabilistic and expected reachability follow in identical fashion.

We begin with a number of preliminary concepts. For a set of vertices V, let $\sqsubseteq$ be the partial order over $(V \to [0,1]) \times (V \to [0,1])$ where $\mathbf{x} \sqsubseteq \mathbf{x}'$ if and only if $x_v \leq x'_v$ for all $v \in V$. For any stochastic game $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$, subset of player 1 vertices V_{done} and target set F, let $F^{lb}_{V_{done}} : (V_1 \to [0,1]) \to (V_1 \to [0,1])$ be the function:

$$F^{lb}_{V_{done}}(\mathbf{x})(v) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } v \in V_1^{lb} \\ 0 & \text{if } v \in V_0^{lb} \\ \inf_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) & \text{if } v \in V_{done} \setminus (V_1^{lb} \cup V_0^{lb}) \\ \min_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot x_v & \text{otherwise} \end{cases}$$

where the notation $v' \leftarrow v$ denotes the set $\{v' \in V \mid (v, v') \in E\}$ and:

$$V_1^{lb} = \{v \in V_1 \mid \inf_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) = 1\}$$

$$V_0^{lb} = \{v \in V_1 \mid \inf_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) = 0\}.$$

Furthermore, let $F^{ub}_{V_{done}} : (V_1 \to [0,1]) \to (V_1 \to [0,1])$ be the function:

$$F^{ub}_{V_{done}}(\mathbf{x})(v) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } v \in V_1^{ub} \\ 0 & \text{if } v \in V_0^{ub} \\ \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) & \text{if } v \in V_{done} \setminus (V_1^{ub} \cup V_0^{ub}) \\ \max_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v \in V} \delta(v_\mathrm{p})(v) \cdot x_v & \text{otherwise} \end{cases}$$

where:

$$V_1^{ub} = \{v \in V_1 \mid \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) = 1\}$$

$$V_0^{ub} = \{v \in V_1 \mid \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) = 0\}.$$

The following results are adapted from [9] to the notation used in this paper.

Lemma 7 *For any stochastic game* G, *target set* F *and subset of player* 1 *vertices* V_{done}, *the function* $F^{lb}_{V_{done}}$ *is monotonic with respect to* $\sqsubseteq$.

Theorem 4 *For any stochastic game* G, *target set* F *and subset of player* 1 *vertices* V_{done}, *the function* $F^{lb}_{V_{done}}$ *has a unique fixed point* $\mathbf{u}^{lb}$ *and* u_v^{lb} *is the optimal value for the probability of reaching* F *when player* 1 *and player* 2 *cooperate in order to minimise it, that is* $u_v^{lb} = \inf_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F)$.

Lemma 8 *For any stochastic game* G, *target set* F *and subset of player* 1 *vertices* V_{done}, *the function* $F^{ub}_{V_{done}}$ *is monotonic with respect to* $\sqsubseteq$.

Theorem 5 *For any stochastic game* G, *target set* F *and subset of player* 1 *vertices* V_{done}, *the function* $F^{ub}_{V_{done}}$ *has a unique fixed point* $\mathbf{u}^{ub}$ *and* u_v^{ub} *is the optimal value for the probability of reaching* F *for player* 1, *that is* $u_v^{ub} = \sup_{\sigma_1} \inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F)$.

Using these results and the Knaster-Tarski theorem, to prove the convergence of the value iteration scheme of Fig. 5, supposing $\mathbf{x}^{lb,init}$ and $\mathbf{x}^{ub,init}$ are the initial values for computing $\mathbf{u}^{lb}$ and $\mathbf{u}^{ub}$ respectively, it is sufficient to show that:

$$\mathbf{x}^{\star,init} \sqsubseteq \mathbf{u}^{\star} \quad \text{and} \quad \mathbf{x}^{\star,init} \sqsubseteq F^{\star}_{V_{done}}(\mathbf{x}^{\star,init}) \quad \text{for } \star \in \{lb, ub\}. \tag{3}$$

Therefore, suppose that P and P^{old} are the partitions from the current and previous refinement steps, $\mathsf{G}^{P}_{\mathsf{M}}$ and $\mathsf{G}^{P^{old}}_{\mathsf{M}}$ are the corresponding abstractions and V_{done} is set of vertices of $\mathsf{G}^{P^{old}}_{\mathsf{M}}$ for which the lower and upper bounds are equal. Note that, by construction we have that $P \preceq P^{old}$. We now prove that (3) holds by considering the cases where $\star = lb$ and $\star = ub$ in turn.

– If $\star = lb$ then by construction, the initial vector $\mathbf{x}^{lb,init}$ is given by:

$$x_v^{lb,init} = \begin{cases} 1 & \text{if } v \in V_1^{lb} \\ 0 & \text{if } v \in V_0^{lb} \\ p_{v^{old}}^{lb,\min}(F) & \text{otherwise} \end{cases}$$

where v^{old} is the unique element of P^{old} such that $v \subseteq v^{old}$. Now, for any $v \in V_1$, by Theorem 2 we have that:

$$p_{v^{old}}^{lb,\min}(F) \leq p_v^{lb,\min}(F) (= u_v^{lb}) \tag{4}$$

and hence $\mathbf{x}^{lb,init} \sqsubseteq \mathbf{u}^{lb}$.

Therefore, it remains to show that $\mathbf{x}^{lb,init} \sqsubseteq F^{lb}_{V_{done}}(\mathbf{x}^{lb,init})$. Now consider any $v \in V_1$. First, by construction we have:

$$p_{v^{old}}^{lb,\min}(F) \leq x_v^{lb,init}. \tag{5}$$

Next, if $v \in V_1^{lb} \cup V_0^{lb} \cup V_{done}$, then by definition of $F^{lb}_{V_{done}}$ we have $F^{lb}_{V_{done}}(\mathbf{x}^{lb,init})(v) = p_v^{lb,\min}(F)$, and hence using (4) we have $F^{lb}_{V_{done}}(\mathbf{x}^{lb,init})(v) \geq x_v^{lb,init}$. On the other hand, if $v \notin V_1^{lb} \cup V_0^{lb} \cup V_{done}$, then by definition of $F^{lb}_{V_{done}}$:

$$
\begin{aligned}
&F^{lb}_{V_{done}}(\mathbf{x}^{lb,init})(v) \\
&= \min_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_\mathrm{p})(v_1) \cdot x_{v_1}^{lb,init} \\
&\geq \min_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_\mathrm{p})(v_1) \cdot p_{v_1^{old}}^{lb,\min}(F) && \text{by (5)} \\
&= \min_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v_1^{old} \in V_1^{old}} \sum_{v_1 \subseteq v_1^{old}} \delta(v_\mathrm{p})(v_1) \cdot p_{v_1^{old}}^{lb,\min}(F) && \text{since } P \preceq P_{old} \\
&= \min_{v_2 \leftarrow v} \min_{v_\mathrm{p} \leftarrow v_2} \sum_{v_1^{old} \in V_1^{old}} \delta_{old}(v_\mathrm{p})(v) \cdot p_{v_1^{old}}^{lb,\min}(F) && \text{by Definition 10} \\
&= \min_{v_2^{old} \leftarrow v^{old}} \min_{v_\mathrm{p}^{old} \leftarrow v_2^{old}} \sum_{v_1^{old} \in V_1^{old}} \delta_{old}(v_\mathrm{p}^{old})(v_1^{old}) \cdot p_{v_1^{old}}^{lb,\min}(F) && \text{since } P \preceq P_{old} \\
&= p_{v^{old}}^{lb,\min}(F) && \text{by Theorem 4} \\
&= x_v^{lb,init} && \text{by construction.}
\end{aligned}
$$

Since these are all the possible cases to consider, we have $\mathbf{x}^{lb,init} \sqsubseteq F^{lb}_{V_{done}}(\mathbf{x}^{lb,init})$ as required.

– If $\star = ub$, then by construction:

$$
x_v^{ub,init} = \begin{cases} 1 & \text{if } v \in V_1^{ub} \\ 0 & \text{if } v \in V_0^{ub} \\ p_v^{ub,\min}(F) & \text{if } v \in V_{done} \setminus (V_1^{ub} \cup V_0^{ub}) \\ p_v^{lb,\min}(F) & \text{otherwise.} \end{cases}
$$

Now for any $v \in V_1$ by definition of $p_v^{lb,\min}(F)$ and $p_v^{ub,\min}(F)$ and construction respectively, we have:

$$
p_v^{lb,\min}(F) \leq p_v^{ub,\min}(F) (= u_v^{ub}) \tag{6}
$$

and hence $\mathbf{x}^{ub,init} \sqsubseteq \mathbf{u}^{ub}$.

Therefore, it remains to show that $\mathbf{x}^{ub,init} \sqsubseteq F^{ub}_{V_{done}}(\mathbf{x}^{ub,init})$. Now consider any $v \in V_1$. First by construction, we have

$$
p_v^{lb,\min}(F) \leq x_v^{lb,init}. \tag{7}
$$

Next, if $v \in V_1^{lb} \cup V_1^{lb} \cup V_{done}$ then by construction $F^{ub}_{V_{done}}(\mathbf{x}^{ub,init})(v) = p_v^{ub,\min}(F)$, and hence using (6) we have $F^{ub}_{V_{done}}(\mathbf{x}^{ub,init})(v) \geq x_v^{ub,init}$. On the other hand, if $v \notin V_1^{lb} \cup V_1^{lb} \cup V_{done}$ then by definition of $F^{ub}_{V_{done}}$:

$$
\begin{aligned}
F^{ub}_{V_{done}}(\mathbf{x}^{ub,init})(v) &= \max_{v_2 \leftarrow v} \min_{v_p \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_p)(v_1) \cdot x^{ub,init}_{v_1} \\
&\geq \max_{v_2 \leftarrow v} \min_{v_p \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_p)(v_1) \cdot p^{lb,\min}_{v_1}(F) && \text{by (7)} \\
&\geq \min_{v_2 \leftarrow v} \min_{v_p \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_p)(v_1) \cdot p^{lb,\min}_{v_1}(F) && \text{rearranging} \\
&= p^{lb,\min}_{v}(F) && \text{by Theorem 5} \\
&= x^{ub,init}_{v} && \text{by construction.}
\end{aligned}
$$

Since these are all the cases to consider, we have $\mathbf{x}^{ub,init} \sqsubseteq F^{ub}_{V_{done}}(\mathbf{x}^{ub,init})$ as required.

This completes the proof of (3), and hence that the value iteration scheme of Fig. 5 converges.

5 Experimental results

We have implemented our abstraction-refinement framework using a prototype Java implementation. This section presents experimental results for the performance of our techniques on case studies from the repository of the probabilistic model checker PRISM [37]. Below are listed the case studies and (in parentheses) the reachability properties used.[1]

- the Zeroconf network configuration protocol for N configured hosts and M IP addresses ("the minimum probability that the host configures correctly");
- the IEEE 802.11 WLAN and IEEE 802.3 CSMA/CD protocols using a backoff counter maximum of bc and a maximum packet send time of 500 µs ("the minimum probability that a station's backoff counter reaches bc").
- the FireWire root contention protocol for a network transmission delay of d ns ("the maximum expected time to elect a leader");
- the randomised consensus shared coin protocol of Aspnes and Herlihy for N processes and parameter K ("the maximum expected time until termination").

Several of the models (FireWire, WLAN and CSMA/CD) were modelled using probabilistic timed automata and translated into MDPs using the digital clocks semantics of [29]. For all experiments, the initial partition P contained: (1) the initial state(s); (2) the target states; and (3) all remaining states. The abstraction was refined until the maximum *relative* difference between the bounds for the initial state(s) was below $\varepsilon = 10^{-4}$, i.e. when

$$\max_{v \in V_{init}} \frac{x^{ub}_v - x^{lb}_v}{x^{ub}_v} < 10^{-4}$$

where V_{init} is the set of abstract states that contain at least one concrete initial state. The value iteration algorithm used the default convergence criteria of PRISM (maximum relative difference less than 10^{-6}).

Table 2 presents statistics for the performance of our implementation. For each example, we give the size of the original MDP (i.e. number of concrete states) and, for each of our two

[1] Supporting files for the models and properties used in the experimental results are available from: http://www.prismmodelchecker.org/files/fmsd-games/.

Table 2 Performance statistics for the abstraction-refinement implementation

Case study (parameter)		States in MDP	Strategy-based refinement			Value-based refinement		
			States	Steps	Time (s)	States	Steps	Time (s)
IPv4 Zeroconf protocol (N M)	4 32	26,121	1,041	206	51.66	699	112	28.27
	4 64	50,377	1,041	189	76.30	676	112	49.86
	4 128	98,889	917	179	127.3	680	112	95.80
	8 32	552,097	2,277	208	1,184	883	128	791.7
	8 64	1,065,569	2,277	188	2,412	877	128	1,584
	8 128	2,092,513	1,720	217	4,293	855	128	3,300
IEEE 802.11 WLAN (bc)	2	28,480	399	70	13.98	253	70	14.50
	3	96,302	1,141	118	84.77	704	118	83.00
	4	345,000	2,619	214	571.9	1,584	214	557.6
	5	1,295,218	5,569	406	4,512	3,605	406	4,625
IEEE 802.3 CSMA/CD (bc)	4	92,978	419	53	30.25	410	53	30.2
	5	277,493	846	78	129.8	839	77	126.9
	6	793,110	1,671	138	678.6	1,663	138	654.3
	7	2,221,189	3,297	266	3,687	3,176	267	3,580
IEEE 1394 FireWire root contention (d)	30	4,093	1,274	320	106.7	910	860	307.1
	60	8,618	2,141	244	181.9	1,495	994	588.0
	120	22,852	3,957	183	276.7	2,746	1,404	1,576
	240	84,152	7,956	203	833.3	5,572	2,379	7,681
Randomised consensus protocol (N K)	5 2	173,056	1,298	47	177.5	566	236	690.1
	5 4	327,936	2,529	76	801.7	1,111	346	1,940
	5 8	637,696	5,008	155	5,548	2,074	557	7,368
	5 16	1,257,216	9,987	282	39,803	4,144	1,027	34,133

refinement techniques, the size of the final abstraction (i.e. number of abstract states/player 1 vertices), the number of refinement steps and the total time required for the entire sequence of refinements and value iteration computations.

The abstractions The first, and most important, conclusion that we draw from these results is the quality of the automatically generated abstractions. For results of a relatively high precision ($\varepsilon = 10^{-4}$, i.e. the bounds differ by less than 4 significant figures), the abstractions yielded state-space reductions of up to four orders of magnitude. We find it very promising that this holds for such a wide range of models.

It is also interesting to compare the abstractions we have obtained with manually developed abstractions derived for the same case studies in the literature. For the FireWire example, employing the manual abstraction process of [42] (4 refinement steps and a number of complex proofs) and the digital clocks semantics of [29] yields an abstract model with 1,212 states for $d = 30$. For the Zeroconf example, the manual abstraction of [28] has 737 ($N = 4$, $M = 64$) and 881 ($N = 8$, $M = 64$) states. For both models we achieve smaller abstractions, in a fully automatic fashion, than those obtained manually. Furthermore, there is scope to combine the approaches, e.g. start with a human-derived abstraction and then refine mechanically.

Table 3 More detailed timing statistics for the abstraction-refinement implementation

Case study (parameter)		Abstraction-refinement time (s)					PRISM time (s)
		Total	Breakdown			Optimisation saving	
			Build	Refine	Solve		
Zeroconf (N M)	8 64	1,584	1,574	2.91	6.41	18.83	149.5
	8 128	3,300	3,287	5.78	7.36	52.47	269.8
WLAN (bc)	4	557.6	527.8	0.33	29.02	153.2	19.34
	5	4,625	4,300	1.763	320.9	158.1	89.80
CSMA (bc)	6	654.3	616.3	3.63	33.91	61.79	41.39
	7	3,580	3,322	15.42	240.6	394.6	134.3
FireWire (d)	120	276.7	32.35	0.10	244.0	267.0	8.71
	240	833.3	151.4	0.23	680.9	841.0	25.37
Consensus (N K)	5 8	7,368	5,637	4.33	1,726	4,027	2,379
	5 16	34,133	21,837	13.28	12,278	41,878	14,956

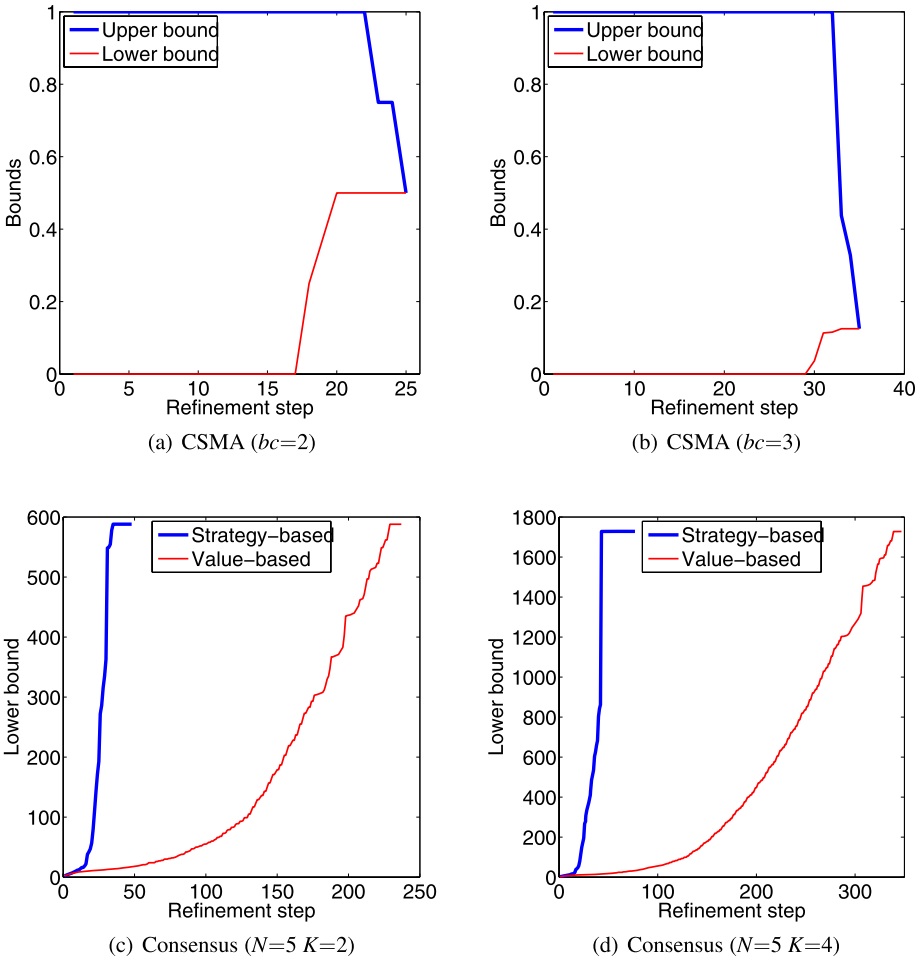
The refinement techniques Value-based refinement mostly outperforms the strategy-based variant, both in terms of the number of refinement steps and the size of the final abstraction. The faster convergence for the value-based approach is likely to be caused by the splitting of partition elements into four, rather than three, at each step. The fact that this is not at the expense of generating larger abstract models suggests that value-based refinement, as intended, avoids splitting states which should stay in the same partition element.

However, on some case studies—FireWire for example—strategy-based refinement requires much fewer steps and is thus quicker. Despite this, the final abstractions produced are slightly larger. Similar results are observed for the consensus case study but, for the largest models, the increased size of the abstraction makes the process slower overall.

Performance The motivations for this work are to establish the viability of the stochastic two-player games as an abstract model for MDPs and to explore the benefits of our abstraction-refinement framework. For convenience, the implementation used in this paper performs model-level abstraction using a prototype abstraction tool which first builds the full concrete MDP (using PRISM) and then reduces it to a game based on a partition of the state space. As a result, this is currently the most time-intensive part of the process. This is illustrated by Table 3 which shows, for the largest models in each case study, a breakdown of the timing for each part of the best-performing refinement technique: abstraction construction ("Build"), refinement ("Refine") and numerical solution using value iteration ("Solve"). Despite this, the overall performance of the framework is very encouraging.

Table 3 also gives the time-savings obtained by using the optimisations from Sect. 4.2 which reuse earlier numerical results to improve convergence. Comparing the time spent on numerical solution (column "Solve") with the time savings achieved (column "Optimisation Saving"), we see that the optimisations result in substantial improvements in performance.

Lastly, Table 3 shows the time required to verify each model in PRISM (using the fastest available engine in the tool). We see that, perhaps unsurprisingly, verification with PRISM is currently faster than using the abstraction-refinement approach. We emphasise that the main goals of this are work are not to improve solution time for instances where it is feasible



(a) CSMA (bc=2) (b) CSMA (bc=3)

(c) Consensus (N=5 K=2) (d) Consensus (N=5 K=4)

Fig. 6 Illustrations of convergence for the CSMA and Consensus case studies

to verify the full concrete model, but rather to develop techniques that will scale to larger models. This will be achieved by building abstractions directly from high-level languages, e.g. using predicate abstraction, and bypassing the construction of the concrete model. Even without this, in some cases (e.g. Zeroconf), the time for abstraction refinement is much faster than PRISM, when the time for explicit construction of the abstraction is disregarded.

Convergence Finally, we look at the convergence of the refinement process, i.e., the changes in difference between the bounds at each refinement step. Previously, we argued that the difference between the bounds provided a useful quantitative measure of the quality of the abstraction. To illustrate this fact, Fig. 6 presents, for two of the CSMA/CD models, the lower and upper bounds on the corresponding probabilistic reachability property after each refinement step when using the value-based refinement method and, for two of the consensus models, the lower bounds for the expected reachability property after each refinement step for both refinement methods. Although there are several sudden drops (or increases) in

these bounds, the overall trends show a gradual convergence in the quantitative results. One reason for the sudden jumps is the fact that the values shown in the graphs in Fig. 6 are for the initial state of each model, whereas the refinement scheme attempts to improve lower and upper the bounds for all states. This means that not every refinement will improve the bounds for the initial state.

6 Related work

Below, we compare our approach with the closest work in the field, namely implementations of abstraction and refinement methods for MDPs. General issues relating to abstraction in the field of probabilistic model checking are discussed in [24, 34].

D'Argenio et al. [11] introduce an approach for verifying reachability properties of MDPs based on probabilistic simulation [38] and a corresponding tool RAPTURE. Properties are analysed on abstractions obtained through successive refinements, starting from a coarse partition derived from the property under study. Since the abstractions used are themselves MDPs, this approach only produces a lower (upper) bound for the minimum (maximum) reachability probability, and hence appears more suited to analysing Markov chains (which contain no nondeterminism and thus the minimum and maximum probabilities coincide). In [11], refinement is repeated until the probability (not the error bound) obtained falls below a given threshold.

Based on [11] and using predicate abstraction [19], Hermanns et al. [21, 44] propose a CEGAR framework for MDPs described in the PRISM language and implement it in a tool called PASS [35]. The CEGAR framework is used for verifying or refuting properties of the form "the maximum probability of error is at most p" for a given probability threshold p. Since abstractions produce only upper bounds on maximum probabilities, to refute a property, probabilistic counterexamples [20] (comprising multiple paths whose combined probability exceeds p) are generated. If these paths are spurious, then they are used to generate further predicates using interpolation.

Perhaps the most important distinguishing feature of our work is the use of two-player games as abstractions. These provide *lower and upper bounds*, avoiding enumeration of a set of paths for a counterexample (which may be large or even infinite, resulting in non-termination). In addition, the bounds enable a *quantitative* approach: we target properties without thresholds such as "what is the maximum probability of failure?". Recent extensions to the PASS tool [43] use the game-based abstraction approach introduced in this paper and demonstrate that this is faster and yields smaller abstractions. Furthermore, the abstraction-refinement framework presented in this paper has been successfully applied in several other contexts. In [25], predicate abstraction is used to construct stochastic game abstractions of PRISM models. In [26] and [30], game-based abstraction and refinement are used to develop verification techniques for probabilistic software and probabilistic timed automata, respectively. In the former, the techniques are used to verify ANSI-C code for network utilities whose complexity is significantly beyond the scope of existing probabilistic verification tools. In the latter, methods to analyse infinite-state real-time systems are presented that outperform all existing approaches.

In [14], de Alfaro et al. propose a technique called 'magnifying-lens abstraction' (MLA) which computes lower and upper bounds on reachability probabilities of MDPs. The approach is based on, first, partitioning the state space into regions and then analysing ('magnifying') the states of each region separately by applying value iteration locally. Regions are refined adaptively until the difference between the bounds for all regions is within some

specified accuracy. MLA is suited to models where there is some notion of 'distance' between states (states close together have similar reachability probabilities) which is used during refinement. Since probabilities are computed for each state of each region and then a minimum/maximum is stored for each region, analysing either small numbers of large regions or large number of small regions is expensive in time and memory. Furthermore, the greatest possible reduction in state-space N is $O(\sqrt{N})$.

Comparing results in [14] for Zeroconf ($N = 4$, $M = 32$), which has 26,121 concrete states, using the same accuracy of $\varepsilon = 10^{-3}$, the MLA approach gives 131 regions and a maximum region size of 1,005, whereas our value-based refinement yields 699 abstract states. Note that, as Fig. 2 demonstrates, our abstract model is smaller for larger values of M while, for MLA, increasing M will increase either the number of regions or maximum region size (or both). Similarly, our framework is applicable to infinite-state systems, whereas [14] is not.

An alternative approach to the abstraction of MDPs is [4], which proposes a CEGAR framework for verifying a fragment of the logic PCTL using MDPs as abstract models (as in [11]). The focus of the paper is on suitable notions of counterexamples for this process, observing that existing proposals such as those used in [21] are not sufficiently expressive for general PCTL formulae. Instead, the paper proposes that counterexamples take the form of "small" MDPs that simulate the concrete model. Algorithms to implement the proposed CEGAR framework are given, but no implementation is attempted.

Elsewhere, abstraction techniques have been considered for discrete-time Markov chains [18, 39], which can be seen as MDPs without nondeterminism. These techniques abstract probabilities to intervals; however, practical applicability has yet to be demonstrated. Also relevant is the work of Chatterjee et al. [6] who propose a counterexample-based abstraction-refinement technique for planning problems on stochastic games. Again, an implementation to test this on practical examples has not yet been developed.

Another important direction for abstraction and refinement of probabilistic systems is the extension of abstract interpretation to the probabilistic setting [33, 36, 41], although these approaches have yet to be combined with refinement. For details on the relationship between the game-based abstraction approach and abstract interpretation, see [43].

In [17] a method for approximating continuous-state (and hence infinite-state) Markov processes by a family of finite-state Markov chains is presented. It is shown that, for simple quantitative modal logic, if the continuous Markov process satisfies a formula, then one of the approximations also satisfies the formula. Finally, McIver and Morgan have developed a framework for the refinement and abstraction of probabilistic programs using expectation transformers [32]. The proof techniques developed in this work have been implemented in the HOL theorem-proving environment [23].

7 Conclusions

We have presented a novel abstraction-refinement framework for Markov decision processes. Our abstraction technique is based on the use of stochastic two-player games, in which one player corresponds to nondeterministic choices from the MDP and the other corresponds to the nondeterminism introduced through abstraction. Using existing results and algorithms from the stochastic games literature, we have shown how the abstraction can be used to compute both lower and upper bounds on the minimum and maximum probability or expected reward of reaching a set of states in the original MDP.

In addition to providing quantitative results for properties of the MDP, the stochastic game yields a measure of the utility of the abstraction, thus giving a basis for refining it to

improve its accuracy. Using this abstraction approach, we have introduced two refinement techniques and an optimised algorithm for a quantitative abstraction-refinement loop. Our experimental results illustrate how this framework provides efficient and fully automatic generation of precise yet compact abstractions for large MDPs from a wide selection of complex case studies. This demonstrates that stochastic two-player games are indeed a good choice of model for representing abstractions of MDPs.

There are many possibilities for future work on this topic. An important direction is to investigate the application of the framework at the level of MDP modelling formalisms. In fact, these techniques have subsequently been employed to develop probabilistic verification methods for imperative languages such as C [26], using predicate abstraction, and for probabilistic timed automata [30], using convex polyhedra [30]. In each case, a key challenge is developing efficient techniques and heuristics to guide refinement; this represents an important area of future work. We also plan to extend our abstraction and refinement methods to other types of models, for example, continuous-time models and hybrid automata.

Acknowledgements This work was supported in part by EPSRC grants EP/D07956X and EP/D076625. The authors would like to thank the anonymous referees for their helpful comments.

References

1. Baier C, Kwiatkowska M (1998) Model checking for a probabilistic branching time logic with fairness. Distrib Comput 11(3):125–155
2. Bertsekas D, Tsitsiklis J (1991) An analysis of stochastic shortest path problems. Math Oper Res 16(3):580–595
3. Billingsley P (1979) Probability and measure. Wiley, New York
4. Chadha R, Viswanathan M (2010) A counterexample guided abstraction-refinement framework for Markov decision processes. ACM Trans Comput Logic (to appear)
5. Chatterjee K, de Alfaro L, Henzinger T (2004) Trading memory for randomness. In: Proc. 1st int. conf. quantitative evaluation of systems (QEST'04). IEEE Comput. Soc., Los Alamitos, pp 206–217
6. Chatterjee K, Henzinger T, Jhala R, Majumdar R (2005) Counterexample-guided planning. In: Proc. 21st conference in uncertainty in artificial intelligence (UAI'05), pp 104–111
7. Cheshire S, Adoba B, Guttman E (2002) Dynamic configuration of IPv4 link-local addresses (draft August 2002). Zeroconf Working Group of the Internet Engineering Task Force (www.zeroconf.org)
8. Clarke E, Grumberg O, Jha S, Lu Y, Veith H (2000) Counterexample-guided abstraction refinement. In: Emerson A, Sistla A (eds) Proc. 12th int. conf. computer aided verification (CAV'00). Lecture notes in computer science, vol 1855. Springer, Berlin, pp 154–169
9. Condon A (1992) The complexity of stochastic games. Inf Comput 96(2):203–224
10. Condon A (1993) On algorithms for simple stochastic games. Advances in computational complexity theory. DIMACS Ser Discrete Math Theor Comput Sci 13:51–73
11. D'Argenio P, Jeannet B, Jensen H, Larsen K (2001) Reachability analysis of probabilistic systems by successive refinements. In: de Alfaro L, Gilmore S (eds) Proc. 1st joint int workshop process algebra and probabilistic methods, performance modelling and verification (PAPM/PROBMIV'01). Lecture notes in computer science, vol 2165. Springer, Berlin, pp 39–56
12. de Alfaro L (1999) Computing minimum and maximum reachability times in probabilistic systems. In: Baeten J, Mauw S (eds) Proc. 10th int. conf. concurrency theory (CONCUR'99). Lecture notes in computer science, vol 1664. Springer, Berlin, pp 66–81
13. de Alfaro L (1997) Formal verification of probabilistic systems. Ph.D. thesis, Stanford University
14. de Alfaro L, Roy P (2007) Magnifying-lens abstraction for Markov decision processes. In: Damm W, Hermanns H (eds) Proc. 19th int. conf. computer aided verification (CAV'07). Lecture notes in computer science, vol 4590. Springer, Berlin, pp 325–338
15. de Alfaro L, Henzinger T, Kupferman O (1998) Concurrent reachability games. In: Proc. 39th symp. foundations of computer science (FOCS'98). IEEE Comput. Soc., Los Alamitos, pp 564–575
16. de Alfaro L, Henzinger T, Kupferman O (2007) Concurrent reachability games. Theor Comput Sci 386(3):188–217
17. Desharnais J, Gupta V, Jagadeesan R, Panangaden P (2003) Approximating labelled Markov processes. Inf Comput 184(1):160–200

18. Fecher H, Leucker M, Wolf V (2006) Don't know in probabilistic systems. In: Valmari A (ed) Proc. 13th int. spin workshop on model checking of software (SPIN'06). Lecture notes in computer science, vol 3925. Springer, Berlin, pp 71–88
19. Graf S, Saidi H (1997) Construction of abstract state graphs with PVS. In: Grumberg O (ed) Proc. 9th int. conf. computer aided verification (CAV'97). Lecture notes in computer science, vol 1254. Springer, Berlin, pp 72–83
20. Han T, Katoen JP, Damman B (2009) Counterexample generation in probabilistic model checking. IEEE Trans Softw Eng 35(2):241–257
21. Hermanns H, Wachter B, Zhang L (2008) Probabilistic CEGAR. In: Gupta A, Malik S (eds) Proc. 20th int. conf. computer aided verification (CAV'08). Lecture notes in computer science, vol 5123. Springer, Berlin, pp 162–175
22. Hinton A, Kwiatkowska M, Norman G, Parker D (2006) PRISM: A tool for automatic verification of probabilistic systems. In: Hermanns H, Palsberg J (eds) Proc. 12th int. conf. tools and algorithms for the construction and analysis of systems (TACAS'06). Lecture notes in computer science, vol 3920. Springer, Berlin, pp 441–444
23. Hurd J, McIver A, Morgan C (2005) Probabilistic guarded commands mechanized in HOL. Theor Comput Sci 346(1):96–112
24. Huth M (2004) An abstraction framework for mixed nondeterministic and probabilistic systems. In: Baier C, Haverkort B, Hermanns H, Katoen JP, Siegle M (eds) Validation of stochastic systems. Lecture notes in computer science, vol 2925. Springer, Berlin, pp 419–444
25. Kattenbelt M, Kwiatkowska M, Norman G, Parker D (2008) Game-based probabilistic predicate abstraction in PRISM. In: Proc. 6th workshop quantitative aspects of programming languages (QAPL'08)
26. Kattenbelt M, Kwiatkowska M, Norman G, Parker D (2009) Abstraction refinement for probabilistic software. In: Jones N, Muller-Olm M (eds) Proc. 10th int. conf. verification, model checking and abstract interpretation (VMCAI'09). Lecture notes in computer science, vol 5403. Springer, Berlin, pp 182–197
27. Kemeny J, Snell J, Knapp A (1976) Denumerable Markov chains, 2nd edn. Springer, Berlin
28. Kwiatkowska M, Norman G, Parker D (2006) Game-based abstraction for Markov decision processes. In: Proc. 3th int. conf. quantitative evaluation of systems (QEST'06). IEEE Comput. Soc., Los Alamitos, pp 157–166
29. Kwiatkowska M, Norman G, Parker D, Sproston J (2006) Performance analysis of probabilistic timed automata using digital clocks. Form Methods Syst Des 29:33–78
30. Kwiatkowska M, Norman G, Parker D (2009) Stochastic games for verification of probabilistic timed automata. In: Ouaknine J, Vaandrager F (eds) Proc. 7th international conference on formal modelling and analysis of timed systems (FORMATS'09). Lecture notes in computer science, vol 5813. Springer, Berlin, pp 212–227
31. Larsen K, Skou A (1991) Bisimulation through probabilistic testing. Inf Comput 94:1–28
32. McIver A, Morgan C (2004) Abstraction, refinement and proof for probabilistic systems. Monographs in computer science. Springer, Berlin
33. Monniaux D (2005) Abstract interpretation of programs as Markov decision processes. Sci Comput Program 58(1–2):179–205
34. Norman G (2004) Analysing randomized distributed algorithms. In: Baier C, Haverkort B, Hermanns H, Katoen JP, Siegle M (eds) Validation of stochastic systems. Lecture notes in computer science, vol 2925. Springer, Berlin, pp 384–418
35. PASS tool homepage. http://depend.cs.uni-sb.de/PASS/
36. Pierro AD, Hankin C, Wiklicky H (2006) Abstract interpretation for worst and average case analysis. In: Reps T, Sagiv M, Bauer J (eds) Program analysis and compilation, theory and practice, essays dedicated to Reinhard Wilhelm on the occasion of his 60th birthday. Lecture notes in computer science, vol 4444. Springer, Berlin, pp 160–174
37. PRISM web site. http://www.prismmodelchecker.org/
38. Segala R (1995) Modelling and verification of randomized distributed real time systems. Ph.D. thesis, Massachusetts Institute of Technology
39. Sen K, Viswanathan M, Agha G (2006) Model-checking Markov chains in the presence of uncertainties. In: Hermanns H, Palsberg J (eds) Proc. 12th int. conf. tools and algorithms for the construction and analysis of systems (TACAS'06). Lecture notes in computer science, vol 3920. Springer, Berlin, pp 394–410
40. Shapley L (1953) Stochastic games. In: Proc. national academy of science, vol 39, pp 1095–1100
41. Smith M (2008) Probabilistic abstract interpretation of imperative programs using truncated normal distributions. In: Aldini A, Baier C (eds) Proc. 6th workshop on quantitative aspects of programming languages (QAPL'08). Electronic notes in theoretical computer science, vol 220(3). Elsevier, Dordrecht, pp 43–59

42. Stoelinga M, Vaandrager F (1999) Root contention in IEEE 1394. In: Katoen JP (ed) Proc. 5th int. AMAST workshop real-time and probabilistic systems (ARTS'99). Lecture notes in computer science, vol 1601. Springer, Berlin, pp 53–74
43. Wachter B, Zhang L (2010) Best probabilistic transformers. In: Barthe G, Hermenegildo M (eds) Proc. 11th int. conf. verification, model checking and abstract interpretation (VMCAI'10). Lecture notes in computer science, vol 5944. Springer, Berlin, pp 362–379
44. Wachter B, Zhang L, Hermanns H (2006) Probabilistic model checking modulo theories. In: Proc. 4th int. conf. quantitative evaluation of systems (QEST'07). IEEE Comput. Soc., Los Alamitos, pp 129–138