Open access • Journal Article • 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

Institutions: University of Oxford, University of Glasgow

Published on: 01 Sep 2010 - Formal Methods

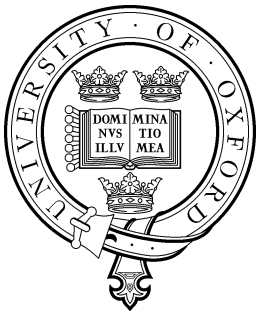
Topics: Abstraction model checking, Abstraction inversion, Partially observable Markov decision process, Markov decision process and Nondeterministic algorithm

Computing Laboratory

A GAME-BASED ABSTRACTION-REFINEMENT FRAMEWORK FOR MARKOV DECISION PROCESSES

Mark Kattenbelt
Marta Kwiatkowska
Gethin Norman
David Parker

CL-RR-08-06



Oxford University Computing Laboratory
Wolfson Building, Parks Road, Oxford OX1 3QD

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 the 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.

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 [13], 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'. Therefore, in this setting a different notion of the correspondence between properties of the concrete and

abstract models is required. A suitable alternative, for 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 cumulated in doing so.

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 abstract MDPs as stochastic two-player games [35, 14], where the two players correspond to the two different forms of nondeterminism. We can then analyse these models using techniques developed for such games [15, 3, 10].

Our analysis of these abstract models results in a separate lower and upper bound for both the minimum and maximum probabilities (or expected reward) 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 [21, 32] 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 [20], which have recently been adapted to probabilistic models [38, 24].

A preliminary version of this paper, introducing the game-based abstraction approach, was published in conference proceedings as [26].

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, Section 4 introduces a corresponding abstraction-refinement framework. In Section 5, we describe a prototype implementation of the techniques from Sections 3 and 4, and present experimental results for a large selection of real-world case studies. Section 6 discusses related work and Section 7 concludes the paper.

2 Background

Let $\mathbb{R}_{\geqslant 0}$ 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 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 1 *A* Markov decision process *is a tuple* $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$*, where*

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

A probabilistic transition $s \xrightarrow{\mu} s'$ is made from a state s by first nondeterministically selecting a distribution $\mu \in \mathit{Steps}(s)$ and then making a probabilistic choice of target state s' according to the distribution μ.

A *path* of an MDP 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 $i \geqslant 0$. We denote by $\pi(i)$ the state s_i and by $\mathit{step}(\pi, i)$ the distribution μ_i. For a finite path π_{fin}, we let $|\pi_{\mathit{fin}}|$ denote the length of π_{fin} (i.e. the number of transitions) and $\mathit{last}(\pi_{\mathit{fin}})$ be its final state. Finally, $\pi^{(i)}$ denotes the prefix of length i of π.

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

Definition 2 *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 is described by a probability space $(Path^A(s), \mathcal{F}_s^A, Prob_s^A)$ over the infinite paths corresponding to A that start in s. This can be defined in standard fashion [25]. Based on this, we introduce two quantitative measures for MDPs which together form the basis for probabilistic model checking of MDPs [1, 7].

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 usual expectation of the function $r(F, \cdot)$ (which returns, for a given path, the total reward accumulated until a state in F is reached along the path) with respect to the probability measure $Prob_s^A$. More precisely:

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

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 which 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.

Quantifying over all adversaries, we consider both the minimum and maximum values of these measures.

Definition 3 *For an MDP $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$, reachability objective $F \subseteq S$ and state $s \in S$, the* minimum and maximum reachability probabilities *of reaching F from s equal:*

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

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

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

Computing values for probabilistic and expected reachability reduces to the *stochastic shortest path problem* for Markov decision processes; see for example [8, 2]. 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 2). A consequence of this is that these quantities can be computed through an iterative processes known as *value iteration* [8, 2], the basis of which is given in the lemma below.

Lemma 2.1 *Consider an MDP* $\mathsf{M} = (S, s_{init}, Steps, \mathbf{r})$ *and set of target states* $F \subseteq S$.

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

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

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

- *The sequences of vectors* $\langle e_n^{\min} \rangle_{n \in \mathbb{N}}$ *and* $\langle e_n^{\max} \rangle_{n \in \mathbb{N}}$ *converge to the minimum and maximum expected reward of reaching the target set* F*, where for any state* $s \in S$*:*
  - *if* $s \in F$*, then* $e_n^{\min}(s) = e_n^{\max}(s) = 0$ *for all* $n \in \mathbb{N}$*;*
  - *if* $s \notin F$*, then:*

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

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

2.2 Stochastic Two-Player Games

In this section, we review (simple) stochastic games [35, 14], which are turn-based games involving two players and chance.

Definition 4 *A* stochastic two-player game *is a tuple* $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_{\mathrm{p}}), \delta, \overline{\mathbf{r}})$ *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*;*
- $\delta : V_\mathrm{p} \to \mathsf{Dist}(V)$ *is the probabilistic transition function;*
- $\overline{\mathbf{r}} : E \to \mathbb{R}_{\geqslant 0}$ *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 operates as follows. Initially, a token is placed on the starting vertex v_{init}. At each step of the game, the token moves from its current vertex v to a neighbouring vertex 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 $v \in V_\mathrm{p}$ then v' is selected randomly according to the distribution $\delta(v)$.

A Markov decision process can 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 the 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 similarly to MDPs [25].

A *reachability objective* of a game G is a set of vertices $\overline{F}$ which a player attempts to reach. For a fixed strategy pair (σ_1, σ_2) and vertex $v \in V$ we define both the probability and expected reward corresponding to the reachability objective $\overline{F}$ as:

$$\begin{aligned} \overline{p}_v^{\sigma_1,\sigma_2}(\overline{F}) &\stackrel{\text{def}}{=} Prob_v^{\sigma_1,\sigma_2}\{\omega \in Play^{\sigma_1,\sigma_2}(v) \mid \exists i \in \mathbb{N} \wedge \omega(i) \in \overline{F}\} \\ \overline{e}_v^{\sigma_1,\sigma_2}(\overline{F}) &\stackrel{\text{def}}{=} \int_{\omega \in Play^{\sigma_1,\sigma_2}(v)} \overline{r}(\overline{F}, \omega)\, \mathrm{d}Prob_v^{\sigma_1,\sigma_2} \end{aligned}$$

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 5 *For a game* $\mathsf{G} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$*, reachability objective* $\overline{F} \subseteq V$ *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:*

$$\sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(\overline{F}) \quad \textit{and} \quad \sup_{\sigma_2} \inf_{\sigma_1} \overline{p}_v^{\sigma_1,\sigma_2}(\overline{F})$$

and the optimal expected rewards are:

$$\sup_{\sigma_1} \inf_{\sigma_2} \overline{e}_v^{\sigma_1,\sigma_2}(\overline{F}) \quad \textit{and} \quad \sup_{\sigma_2} \inf_{\sigma_1} \overline{e}_v^{\sigma_1,\sigma_2}(\overline{F}) \,.$$

A player 1 strategy σ_1 is optimal from vertex v with respect to the probability of the objective if:

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

The optimal strategies for player 2 and for expected rewards can be defined analogously. We now summarise a number of results from [10, 14, 15].

Definition 6 *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 $last(\omega_{fin}) = last(\omega'_{fin})$.*

Similarly to MDPs, for any stochastic game, the family of pure memoryless strategies suffices for optimality with respect to reachability objectives.

Lemma 2.2 *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 $\langle \overline{p}_n \rangle_{n\in\mathbb{N}}$ converges to the optimal probabilities for player 1 with respect to the reachability objective $\overline{F}$, where for any vertex $v \in V$:*
  - *if $v \in \overline{F}$, then $\overline{p}_n(v) = 1$ for all $n \in \mathbb{N}$;*
  - *and otherwise:*

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

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

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

Lemma 2.2 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 2.1.

3 Abstraction for Markov Decision Processes

We now present our notion of abstraction for MDPs. As described in Section 1, the abstract version of a concrete MDP takes the form of a stochastic two-player game where the choices made by player 2 correspond to the nondeterminism in the original MDP and the choices made by player 1 correspond to the nondeterminism introduced by the abstraction process. An *abstract MDP* is a stochastic two-player game, defined by an MDP $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$ and 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, player 1 vertices in the game are abstract states and player 2 vertices correspond to the sets of nondeterministic choices in individual concrete states.

In the following, 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 $S_i \in P$. Similarly, we denote by $\overline{\mathit{Steps}(s)}$ the set $\{(\mathbf{r}(s,\mu), \overline{\mu}) \,|\, \mu \in \mathit{Steps}(s)\}$. 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 [26] where, to simplify the presentation, we assumed that distributions that were lifted to the same abstract distribution had identical reward values.

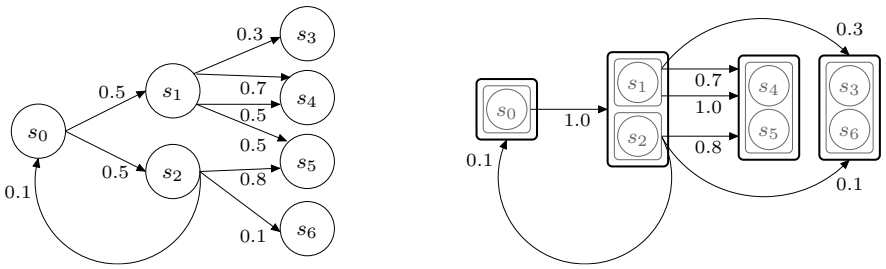
Definition 7 *Given an MDP* $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$ *and a partition* P *of the state space* S*, we define the corresponding* abstract MDP *as the stochastic game*

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

in which:

- $V_1 = P$*;*
- $V_2 = \{v_2 \subseteq \mathbb{R}_{\geqslant 0} \times \mathsf{Dist}(P) \,|\, v_2 = \overline{\mathit{Steps}(s)} \text{ for some } s \in S\}$*;*
- $V_{\mathrm{p}} = \{v_{\mathrm{p}} \in \mathbb{R}_{\geqslant 0} \times \mathsf{Dist}(P) \,|\, v_{\mathrm{p}} \in \overline{\mathit{Steps}(s)} \text{ for some } s \in S\}$*;*
- $v_{init} = S_i$ *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* $\overline{\mu}$*;*
- $\overline{\mathbf{r}} : E \to \mathbb{R}_{\geqslant 0}$ *is the reward function such that for any* $(v, v') \in V \times V$*:*

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



(a) Concrete model (MDP) (b) Abstraction (stochastic game)

Figure 1: Simple example of the abstraction process for Markov decision processes

Example 3.1 *We illustrate the abstraction process on a simple MDP, shown in Figure 1(a) and the partition* $P = \{\{s_0\}, \{s_1, s_2\}, \{s_4, s_5\}, \{s_3, s_6\}\}$*. The abstract MDP is given in Figure 1(b): shapes outlined in thick black lines are player 1 vertices and each one's successor player 2 vertices are outlined with grey lines and enclosed within it. Successor probabilistic vertices (distributions over player 1 vertices) are represented by sets of grouped, labelled arrows emanating from each player 2 vertex. The states of the original MDP, sets of which comprise player 1 and player 2 vertices, are also depicted.*

Intuitively, the roles of the vertices and players in the abstract MDP can be understood as follows. A player 1 vertex corresponds to an abstract state: an element of the partition 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, and Example 3.1 (see Figure 1), perhaps give the impression that the abstraction does not reduce the size of the model. In fact this is generally 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 outgoing distributions and reward values are collapsed into one player 2 state. In fact, there is an even greater reduction since these outgoing distributions have now been lifted to the (smaller) abstract state space. Furthermore, in practice there is no need to store the entire vertex set V of the abstract MDP. 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 3.5 and Section 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 Abstract MDP

In this section we describe how, for an MDP M and state partition P, the abstract MDP G^P_M yields lower and upper bounds for probabilistic reachability and expected reachability properties of M.

We also show that a finer partition results in a more precise abstraction, i.e. an abstract MDP for which the lower and upper bounds are tighter. The notion of a finer partition is defined as follows.

Definition 8 *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.*

To ease notation we let, for $x \in \{p, e\}$:

$$\begin{aligned}
\overline{x}_v^{lb,\min}(\overline{F}) &\stackrel{\text{def}}{=} \inf_{\sigma_1,\sigma_2} \overline{x}_v^{\sigma_1,\sigma_2}(\overline{F}) \\
\overline{x}_v^{ub,\min}(\overline{F}) &\stackrel{\text{def}}{=} \sup_{\sigma_1} \inf_{\sigma_2} \overline{x}_v^{\sigma_1,\sigma_2}(\overline{F}) \\
\overline{x}_v^{lb,\max}(\overline{F}) &\stackrel{\text{def}}{=} \sup_{\sigma_2} \inf_{\sigma_1} \overline{x}_v^{\sigma_1,\sigma_2}(\overline{F}) \\
\overline{x}_v^{ub,\max}(\overline{F}) &\stackrel{\text{def}}{=} \sup_{\sigma_1,\sigma_2} \overline{x}_v^{\sigma_1,\sigma_2}(\overline{F})
\end{aligned}$$

The values $\overline{x}_v^{ub,\min}(\overline{F})$ and $\overline{x}_v^{lb,\max}(\overline{F})$, 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 2.2, these optimal values can be computed using an iterative process. This process can also be used to generate a pair of (memoryless) strategies that result in these optimal values [15]. The other values, $\overline{x}_v^{lb,\min}(\overline{F})$ and $\overline{x}_v^{ub,\max}(\overline{F})$, although not usually considered for games (because the players co-operate), can be computed similarly by considering the game as an MDP [8] (see Lemma 2.1).

The following theorem demonstrates that a finer partition results in a more precise abstraction, i.e. an abstract MDP for which the lower and upper bounds are tighter.

Theorem 3.2 *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 $\mathsf{G}_{\mathsf{M}}^{P} = ((V, E), v_{init}, (V_1, V_2, V_\mathrm{p}), \delta, \overline{\mathbf{r}})$ and $\mathsf{G}_{\mathsf{M}}^{P'} = ((V', E'), v'_{init}, (V_1', V_2', V_\mathrm{p}'), \delta', \overline{\mathbf{r}}')$ are the stochastic games constructed from P and P' according to Definition 7, then for any $x \in \{e, p\}$ and $v \in P$:*

$$\begin{aligned}
\overline{x}_{v'}^{lb,\min}(\overline{F}) &\leqslant \overline{x}_v^{lb,\min}(\overline{F}) \qquad \textit{and} \qquad \overline{x}_v^{ub,\min}(\overline{F}) \leqslant \overline{x}_{v'}^{ub,\min}(\overline{F}) \\
\overline{x}_{v'}^{lb,\max}(\overline{F}) &\leqslant \overline{x}_v^{lb,\max}(\overline{F}) \qquad \textit{and} \qquad \overline{x}_v^{ub,\max}(\overline{F}) \leqslant \overline{x}_{v'}^{ub,\max}(\overline{F})
\end{aligned}$$

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

The following theorem shows how an analysis of the abstract MDP $\mathsf{G}_{\mathsf{M}}^{P'}$ yields lower and upper bounds for probabilistic reachability and expected reachability properties of the MDP M from which it was constructed.

Theorem 3.3 *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 $\mathsf{G}_{\mathsf{M}}^{P'} = ((V', E'), v'_{init}, (V_1', V_2', V_\mathrm{p}'), \delta', \overline{\mathbf{r}}')$ is the stochastic game constructed from M and P' according to Definition 7, then for any $x \in \{e, p\}$ and $s \in S$:*

$$\begin{aligned}
\overline{x}_{v'}^{lb,\min}(\overline{F}) &\leqslant x_s^{\min}(F) \leqslant \overline{x}_{v'}^{ub,\min}(\overline{F}) \\
\overline{x}_{v'}^{lb,\max}(\overline{F}) &\leqslant x_s^{\max}(F) \leqslant \overline{x}_{v'}^{ub,\max}(\overline{F})
\end{aligned}$$

where v' is the unique vertex of V_1' such that $s \in v'$ 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 3.2 and 3.3.

Example 3.4 *Let us return to the previous simple example (see Example 3.1 and Figure 1). Suppose that we are interested in the probability in the original MDP (Figure 1(a)) of, starting from state* s_0*, reaching the target set* $\{s_4, s_5\}$*. The minimum and maximum reachability probabilities can be computed as* $15/19$ (0.789473) *and* $18/19$ (0.947368) *respectively. From the abstraction shown in Figure 1(b) and the results of Theorem 3.3, we can establish that the minimum and maximum probabilities lie within the intervals* $[7/10, 8/9]$ $([0.7, 0.888889])$ *and* $[8/9, 1]$ $([0.888889, 1])$ *respectively.*

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 and maximum reachability probabilities both lay in the interval $[0.7, 1]$*.*

Example 3.5 *To illustrate the application of our abstraction to a larger MDP, we consider a more complex example: a model of the Zeroconf protocol [12] 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 [26].*

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, Figure 2 illustrates results for the minimum probability that the new host configures successfully by time* T*.*

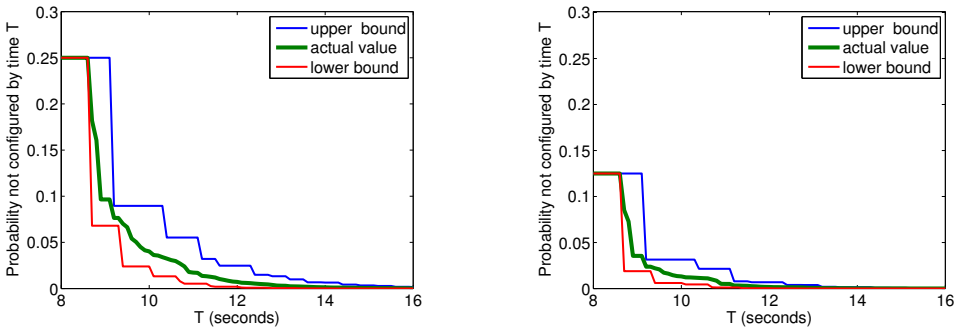
As stated previously, an advantage of our approach is the ability to quantify the utility of the abstraction, based on the difference between the lower and upper bounds obtained. In the case of the plots in Figure 2, for a particular time bound T *this difference is indicated by the vertical distance 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, 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 3.2 and 3.3, presented in Section 3.1. Before doing so, we require a number of preliminary concepts. For the remainder of this section we fix

N	M	States (transitions)		Minimum expected time			Maximum expected time		
		Concrete system	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

Table 1: Zeroconf protocol (Example 3.5): model statistics and numerical results (expected completion time)



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

Figure 2: Zeroconf protocol (Example 3.5): minimum probability the new host configures successfully by time T

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 $\mathsf{G}_{\mathsf{M}}^{P} = ((V, E), v_{init}, (V_1, V_2, V_{\mathrm{p}}), \delta, \overline{\mathbf{r}})$ and $\mathsf{G}_{\mathsf{M}}^{P'} = ((V', E'), v'_{init}, (V_1', V_2', V_{\mathrm{p}}'), \delta', \overline{\mathbf{r}}')$ be the abstract MDPs obtained from P and P' according to Definition 7 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 $\mathsf{G}_{\mathsf{M}}^{P}$ to the vertices of $\mathsf{G}_{\mathsf{M}}^{P'}$.

Definition 9 *Let $[\cdot] : V \to V'$ be the mapping from vertices of $\mathsf{G}_{\mathsf{M}}^{P}$ to vertices of $\mathsf{G}_{\mathsf{M}}^{P'}$ 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* $r \in \mathbb{R}_{\geqslant 0}$ *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 $\omega = v_0 v_1 v_2 \dots$ of $\mathsf{G}^P_{\mathsf{M}}$ we have $[\omega] = [v_0][v_1][v_2]\dots$ 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 the 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 3.6 *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 9 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 $\vec{\sigma} = (\sigma_1, \sigma_2)$ of the game $\mathsf{G}^P_{\mathsf{M}}$, we now construct a (randomised) strategy pair $[\vec{\sigma}]_v = ([\sigma_1]_v, [\sigma_2]_v)$ of the game $\mathsf{G}^{P'}_{\mathsf{M}}$ such that, under $\vec{\sigma}$ and $[\vec{\sigma}]_v$, when starting at v and $[v]$ respectively, the probability of reaching F is equal. For any finite play ω' of $\mathsf{G}^{P'}_{\mathsf{M}}$ such that $\omega'(0) = [v]$ and $\mathit{last}(\omega') \in V'_i$, let the probability of $[\sigma_i]_v$ selecting vertex $v' \in V'$ after play ω' equal:

$$[\sigma_i]_v(\omega')(v') \quad \stackrel{\text{def}}{=} \quad \frac{\mathit{Prob}^{\vec{\sigma}}_v([\omega' v']^{-1}_v)}{\mathit{Prob}^{\vec{\sigma}}_v([\omega']^{-1}_v)}$$

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

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

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

Proposition 3.7 *For any vertex* v *and strategy pair* $\vec{\sigma} = (\sigma_1, \sigma_2)$ *of* $\mathsf{G}^P_{\mathsf{M}}$*, if* Ω *is a measurable set of plays of* $\mathit{Play}^{[\vec{\sigma}]_v}([v])$*, then* $\mathit{Prob}^{[\vec{\sigma}]_v}_{[v]}(\Omega) = \mathit{Prob}^{\vec{\sigma}}_v([\Omega]^{-1}_v)$.

Proof. From the measure construction (see [25]) it is sufficient to show that $\mathit{Prob}^{[\vec{\sigma}]_v}_{[v]}(\omega') = \mathit{Prob}^{\vec{\sigma}}_v[\omega']^{-1}_v$ for any finite play ω' of $\mathsf{G}^{P'}_{\mathsf{M}}$ 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 $[\omega']^{-1}_v = \{v\}$, by the measure construction [25]:

$$\mathit{Prob}^{[\vec{\sigma}]_v}_{[v]}(\omega') = \mathit{Prob}^{\vec{\sigma}}_v[\omega']^{-1}_v = 1\,.$$

Next, suppose by induction that the lemma holds for all plays of length n. Consider any play ω' of length $n{+}1$ of $\mathsf{G}^{P'}_{\mathsf{M}}$. 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 $\mathit{last}(\tilde{\omega}') \in V'_i$ (for $i = 1, 2$), then from measure construction (see [25]):

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

On the other hand, if $last(\tilde{\omega}') \in {V_{\mathrm{p}}}'$, then $last(\tilde{\omega}') = (r, \overline{\mu_{P'}})$ for some $r \in \mathbb{R}_{\geqslant 0}$ and $\mu \in \mathsf{Dist}(S)$, and hence in this case, from the measure construction [25] we have:

$$\begin{aligned}
Prob_{[v]}^{[\vec{\sigma}]_v}(\omega') &= Prob_{[v]}^{[\vec{\sigma}]_v}(\tilde{\omega}') \cdot \overline{\mu_{P'}}(\tilde{v}') \\
&= Prob_v^{\vec{\sigma}}([\tilde{\omega}']_v^{-1}) \cdot \overline{\mu_{P'}}(\tilde{v}') && \text{by induction} \\
&= Prob_v^{\vec{\sigma}}([\tilde{\omega}']_v^{-1}) \cdot \left(\sum\nolimits_{\tilde{v} \in [\tilde{v}']^{-1}} \overline{\mu_P}(\tilde{v}) \right) && \text{by Definition 9} \\
&= \sum\nolimits_{\tilde{v} \in [\tilde{v}']^{-1}} Prob_v^{\vec{\sigma}}([\tilde{\omega}']_v^{-1}) \cdot \overline{\mu_P}(\tilde{v}) && \text{rearranging} \\
&= \sum\nolimits_{\tilde{v} \in [\tilde{v}']^{-1}} Prob_v^{\vec{\sigma}}([\tilde{\omega}']_v^{-1}\tilde{v}) && \text{by definition of } Prob_v^{\vec{\sigma}} \\
&= Prob_v^{\vec{\sigma}}([\omega']_v^{-1}) && \text{by Definition 9 and since } \omega' = \tilde{\omega}'\tilde{v}'
\end{aligned}$$

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

Finally, we require the following lemma and classical result from measure theory.

Lemma 3.8 *For any vertex v and strategy pair $\vec{\sigma} = (\sigma_1, \sigma_2)$ of $\mathsf{G}_{\mathsf{M}}^P$, the mapping $[\cdot]$ of Definition 9 is a measurable function between $(Play^{\vec{\sigma}}(v), \mathcal{F}_v^{\vec{\sigma}})$ and $(Play^{[\vec{\sigma}]_v}([v]), \mathcal{F}_{[v]}^{[\vec{\sigma}]_v})$.*

Proof. The proof follows from measure construction (see [25]). □

Theorem 3.9 ([9]) *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:*

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

We are now in a position to give the proofs of Theorems 3.2 and 3.3.

Proof (of Theorem 3.2). The proof can be reduced to showing that for any pair of strategies $\vec{\sigma} = (\sigma_1, \sigma_2)$ and player 1 vertex v of $\mathsf{G}_{\mathsf{M}}^P$:

$$\overline{p}_v^{\vec{\sigma}}(\overline{F}) = \overline{p}_{[v]}^{[\vec{\sigma}]_v}(\overline{F}) \quad \text{and} \quad \overline{e}_v^{\vec{\sigma}}(\overline{F}) = \overline{e}_{[v]}^{[\vec{\sigma}]_v}(\overline{F})$$

where $[\vec{\sigma}]_v$ is the strategy pair of the game $\mathsf{G}_{\mathsf{M}}^{P'}$ constructed from $\vec{\sigma}$ and v as described above. The first equation follows from Proposition 3.7 using standard results from measure theory. For the second equation, by definition of the expected rewards for games

(see Section 2.2), we have:

$$\begin{aligned}
\overline{e}_v^{\vec{\sigma}}(\overline{F}) &= \int_{\omega \in Play^{\vec{\sigma}}(v)} \overline{\mathbf{r}}(\overline{F}, \omega)\ \mathrm{d}Prob_v^{\vec{\sigma}} \\
&= \int_{\omega \in Play^{\vec{\sigma}}(v)} \overline{\mathbf{r}}'(\overline{F}, [\omega])\ \mathrm{d}Prob_v^{\vec{\sigma}} && \text{by Lemma 3.6} \\
&= \int_{\omega' \in Play^{[\vec{\sigma}]_v}([v])} \overline{\mathbf{r}}(\overline{F}, \omega')\ \mathrm{d}Prob_v^{\vec{\sigma}}([\cdot]^{-1}) && \text{by Theorem 3.9 and Lemma 3.8} \\
&= \int_{\omega' \in Play^{[\vec{\sigma}]_v}([v])} \overline{\mathbf{r}}(\overline{F}, \omega')\ \mathrm{d}Prob_{[v]}^{[\vec{\sigma}]_v} && \text{by Proposition 3.7} \\
&= e_{[v]}^{[\vec{\sigma}]_v}(\overline{F})
\end{aligned}$$

as required. □

Proof (of Theorem 3.3). The proof follows from Theorem 3.2 together with the following facts.

- The partition $P = \{\{s\} \mid s \in S\}$ is finer than all other partitions.
- If $\mathsf{G}_{\mathsf{M}}^P$ is constructed from partition $P = \{\{s\} \mid s \in S\}$ according to Definition 7, then:

$$\begin{aligned}
\overline{x}_{\{s\}}^{lb,\min}(\overline{F}) &= x_s^{\min}(F) = \overline{x}_{\{s\}}^{ub,\min}(\overline{F}) \\
\overline{x}_{\{s\}}^{lb,\max}(\overline{F}) &= x_s^{\max}(F) = \overline{x}_{\{s\}}^{ub,\max}(\overline{F})
\end{aligned}$$

  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 automatic abstraction-refinement of MDPs which uses the game-based abstraction method of Section 3. Unlike in the non-probabilistic case, probabilistic verification is typically quantitative in nature: probabilistic model checking of MDPs essentially involves computing minimum or maximum probabilistic or expected reachability measures. 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 tolerated error ε), then it is desirable to refine the abstraction to reduce the difference. This provides the basis for a quantitative analogue of the counterexample-guided abstraction-refinement process, illustrated in Figure 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.

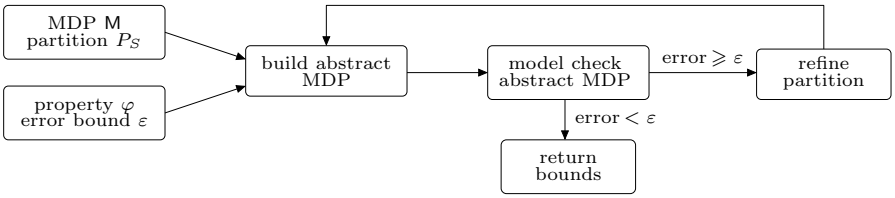


Figure 3: Abstraction-refinement framework for MDPs

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 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 is analogous to the non-probabilistic case where refinement is based on a single counterexample). 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. This guarantees that, for finite state models, the process eventually converges (results in more accurate abstractions). In addition, if a partition is coarser than *probabilistic bisimulation* [28], 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 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 the convergence of the refinement process 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 $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$, 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 $\overline{p}_v^{\,lb,\min}(\overline{F}) < \overline{p}_v^{\,ub,\min}(\overline{F})$ for some $v \in V_1$. The cases for maximum probabilistic and expected reachability follow in identical fashion.

Strategy-based Refinement As mentioned in Section 2, when computing $\overline{p}_v^{\,lb,\min}(\overline{F})$ and $\overline{p}_v^{\,ub,\min}(\overline{F})$, 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 $v \in V$:

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

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:

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

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 4.1 *If there exists a vertex $v' \in V_1$ such that $\overline{p}_{v'}^{\,lb,\min}(\overline{F}) < \overline{p}_{v'}^{\,ub,\min}(\overline{F})$ 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 $\sigma_1^{lb}(v) \neq \sigma_1^{ub}(v)$.*

Proof. The proof is by contradiction. Therefore, suppose that there exists $v' \in V_1$ such that $\overline{p}_{v'}^{\,lb,\min}(\overline{F}) < \overline{p}_{v'}^{\,ub,\min}(\overline{F})$ and $(\sigma_1^{lb}, \sigma_2^{lb})$ and $(\sigma_1^{ub}, \sigma_2^{ub})$ are memoryless strategy pairs such that:

$$\overline{p}_v^{\sigma_1^{lb},\sigma_2^{lb}}(\overline{F}) = \overline{p}_v^{\,lb,\min}(\overline{F}), \;\; \overline{p}_v^{\sigma_1^{ub},\sigma_2^{ub}}(\overline{F}) = \overline{p}_v^{\,ub,\min}(\overline{F}) \;\text{ and } \sigma_1^{lb}(v) = \sigma_1^{ub}(v) \text{ for all } v \in V_1.$$

Now, by the hypothesis we have:

$$\begin{aligned}
\overline{p}_{v'}^{\,lb,\min}(\overline{F}) &< \overline{p}_{v'}^{\,ub,\min}(\overline{F}) \\
&= \sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_{v'}^{\sigma_1,\sigma_2}(\overline{F}) && \text{by definition} \\
&= \inf_{\sigma_2} \overline{p}_{v'}^{\sigma_1^{ub},\sigma_2}(\overline{F}) && \text{since } \sigma_1^{ub} \text{ is an optimal strategy} \\
&\leqslant \overline{p}_{v'}^{\sigma_1^{ub},\sigma_2^{lb}}(\overline{F}) && \text{since } \sigma_2^{lb} \text{ is a player 2 strategy} \\
&= \overline{p}_{v'}^{\sigma_1^{lb},\sigma_2^{lb}}(\overline{F}) && \text{since } \sigma_1^{lb}(v') = \sigma_1^{ub}(v) \text{ for all } v \in V_1 \\
&= \overline{p}_{v'}^{\,lb,\min}(\overline{F}) && \text{by construction of } (\sigma_1^{lb}, \sigma_2^{lb})
\end{aligned}$$

which is a contradiction. □

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 which achieve one of the bounds when refining the partition. More precisely, letting $v_2(s) = \overline{\mathit{Steps}(s)}$, we refine each element $v \in P$ as follows.

1. Construct the following subsets of v:

$$\begin{aligned}
v_{lb}^{\min} &= \{ s \in v \,|\, \overline{p}_{v_2(s)}^{\,lb,\min}(\overline{F}) = \overline{p}_v^{\,lb,\min}(\overline{F}) \} \\
v_{ub}^{\min} &= \{ s \in v \,|\, \overline{p}_{v_2(s)}^{\,ub,\min}(\overline{F}) = \overline{p}_v^{\,ub,\min}(\overline{F}) \}.
\end{aligned}$$

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.

Lemma 4.2 *If there exists $v' \in V_1$ such that $\overline{p}_{v'}^{\,lb,\min}(\overline{F}) < \overline{p}_{v'}^{\,ub,\min}(\overline{F})$, 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 $\overline{p}_{v'}^{\,lb,\min}(\overline{F}) < \overline{p}_{v'}^{\,ub,\min}(\overline{F})$ 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:

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

Now, for any player 1 vertex $v \in V_1$, by definition we have:

$$\begin{aligned} v_{lb}^{\min} &= \{s \in v \,|\, \overline{p}_{v_2(s)}^{\,lb,\min}(\overline{F}) = \overline{p}_v^{\,lb,\min}(\overline{F})\} \\ v_{ub}^{\min} &= \{s \in v \,|\, \overline{p}_{v_2(s)}^{\,ub,\min}(\overline{F}) = \overline{p}_v^{\,ub,\min}(\overline{F})\} \end{aligned}$$

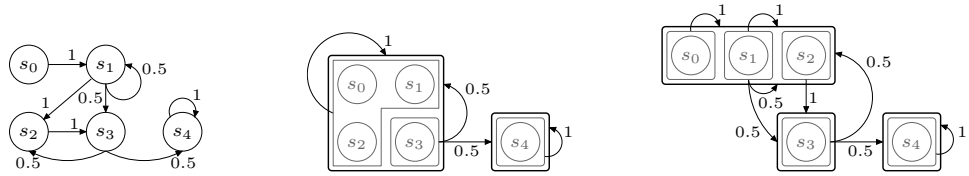
where $v_2(s) = \overline{\mathit{Steps}(s)}$, and hence, by construction $\sigma_1^{lb}(v) = v_2(s_{lb})$ for some $s_{lb} \in v_{lb}^{\min}$ and $\sigma_1^{ub}(v) = v_2(s_{ub})$ for some $s_{ub} \in v_{ub}^{\min}$. Now, by the hypothesis we have $v_{lb}^{\min} = v_{ub}^{\min}$ for all $v \in V_1$ and therefore, since σ_2^{lb} is optimal, it follows that for the player 1 vertex v':

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

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

Example 4.3 *To illustrate the refinement strategies, we consider the simple MDP given in Figure 4(a) with a reward of 1 associated with each transition and 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 game shown in Figure 4(b). As in Example 3.1, shapes outlined in thick black lines are player 1 vertices and each one's successor player 2 vertices are outlined with grey lines and enclosed within it. Successor probabilistic vertices (distributions over player 1 vertices) are represented by sets of grouped, labelled arrows emanating from the vertex. 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 Figure 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



(a) Concrete MDP (b) Initial abstraction (c) Refined abstraction

Figure 4: Example of the abstraction-refinement process

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 Figure 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 (induced) 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* $\{s_0, s_1\}, \{s_2\}, \{s_3\}, \{s_4\}$*.*

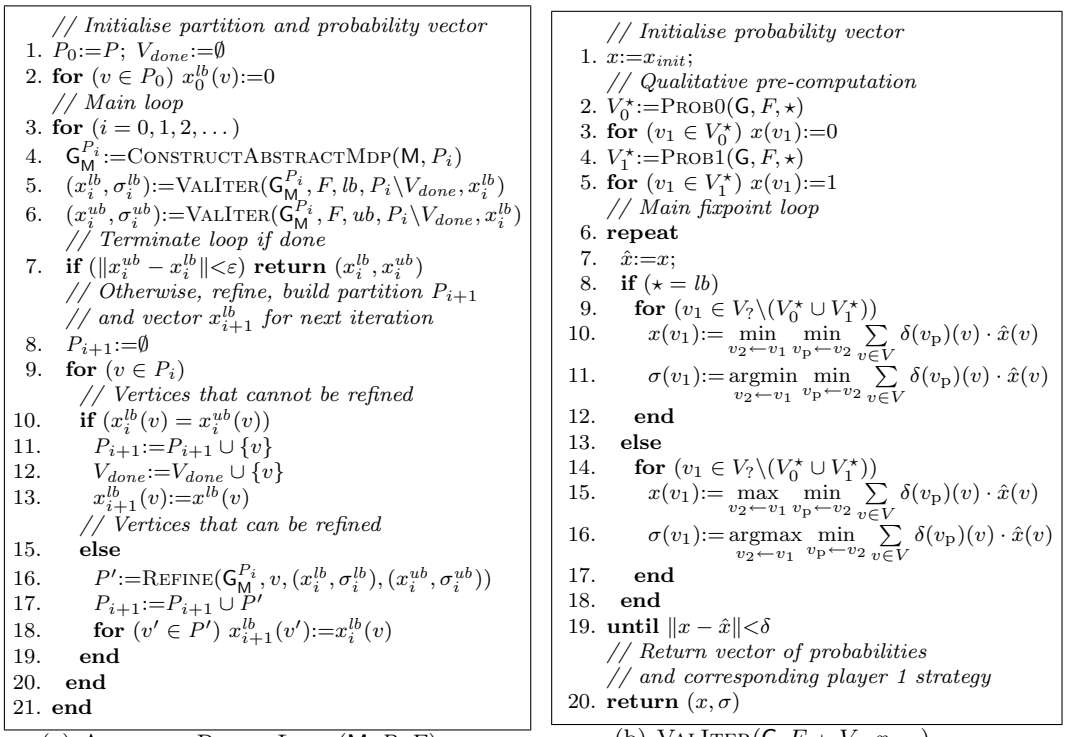
4.2 The Abstraction-Refinement Algorithm

With the refinement strategies of the previous section, we can implement the abstraction-refinement loop shown in Figure 3. Starting with a coarse partition, we iteratively: construct an abstraction (according to Definition 7); compute lower and upper bounds (and corresponding strategies); and refine using either the strategy-based or value-based method of Section 4.1.

In this section, we present an optimised algorithm for the refinement loop, illustrated in Figure 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 x_i^{lb} and x_i^{ub} (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



(a) AbstractRefineLoop(M, P, F)

(b) ValIter($\mathsf{G}, F, \star, V_?, x_{init}$)

Figure 5: Abstraction-refinement loop algorithm (minimum reachability probabilities)

thus have the exact solution for their concrete states); again, we use these values in initial solutions for value iteration.

Numerical solution of the game at each step is performed with an improved version of the standard value iteration algorithm [15], illustrated in Figure 5(b), which includes several optimisations. Firstly, we employ the qualitative algorithms of [4] 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 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 Section 4.3 below.

As in the previous section, the code presented in Figure 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 Figure 5(a) can be either the strategy-based or value-based refinement from Section 4.1. For the latter, we require that lines 11 and 16 of the ValIter algorithm in Figure 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 Figure 5 converges. For simplicity we only consider computing bounds on minimum reachability probabilities and assume a fixed MDP $\mathsf{M} = (S, s_{init}, \mathit{Steps}, \mathbf{r})$ 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 $x \sqsubseteq x'$ if and only if $x(v) \leqslant x'(v)$ 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 reachability objective F, let $F^{lb}_{V_{done}} : (V_1 \to [0,1]) \to (V_1 \to [0,1])$ be the function:

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

where

$$\begin{array}{rcl} V_1^{lb} & = & \{v \in V_1 \mid \inf_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F) = 1\} \\ V_0^{lb} & = & \{v \in V_1 \mid \inf_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F) = 0\} \end{array}$$

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

$$F^{ub}_{V_{done}}(x)(v) \stackrel{\text{def}}{=} \begin{cases} 1 & \text{if } v \in V_1^{ub} \\ 0 & \text{if } v \in V_0^{ub} \\ \sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F) & \text{if } v \in V_{done} \setminus (V_1^{ub} \cup V_0^{ub}) \\ \max\limits_{v_2 \leftarrow v} \min\limits_{v_\mathrm{p} \leftarrow v_2} \sum\limits_{v \in V} \delta(v_\mathrm{p})(v) \cdot x(v) & \text{otherwise} \end{cases}$$

where

$$\begin{array}{rcl} V_1^{ub} & = & \{v \in V_1 \mid \sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F) = 1\} \\ V_0^{ub} & = & \{v \in V_1 \mid \sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F) = 0\}. \end{array}$$

The following results are adapted from [14] to the notation used in this paper.

Lemma 4.4 *For any stochastic game G, objective F and subset of player 1 vertices V_{done}, the function $F^{lb}_{V_{done}}$ is monotonic with respect to $\sqsubseteq$.*

Theorem 4.5 *For any stochastic game G, objective F and subset of player 1 vertices V_{done}, the function $F^{lb}_{V_{done}}$ has a unique fixed point u^{lb} and $u^{lb}(v)$ is the optimal value of the probabilistic reachability objective F when player 1 and player 2 cooperate to minimise the objective, that is $u^{lb}(v) = \inf_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F)$.*

Lemma 4.6 *For any stochastic game G, objective F and subset of player 1 vertices V_{done}, the function $F^{ub}_{V_{done}}$ is monotonic with respect to $\sqsubseteq$.*

Theorem 4.7 *For any stochastic game G, objective F and subset of player 1 vertices V_{done}, the function $F^{ub}_{V_{done}}$ has a unique fixed point u^{ub} and $u^{ub}(v)$ is the optimal value of the probabilistic reachability objective F for player 1, that is $u^{ub}(v) = \sup_{\sigma_1} \inf_{\sigma_2} \overline{p}_v^{\sigma_1,\sigma_2}(F)$.*

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

$$x^{\star}_{init} \sqsubseteq u^{\star} \text{ and } x^{\star}_{init} \sqsubseteq F^{\star}_{V_{done}}(x^{\star}_{init}) \text{ for } \star \in \{lb, ub\}. \tag{2}$$

Therefore, suppose that P and P^{old} are the partitions from the current and previous refinement steps, $\mathsf{G}^{P}_{\mathsf{M}}$ and $\mathsf{G}^{P^{old}}_{\mathsf{M}}$ the corresponding abstract MDPs 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 (2) holds by considering the cases where $\star = lb$ and $\star = ub$ in turn.

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

$$x^{lb}_{init}(v) = \begin{cases} 1 & \text{if } v \in V^{lb}_1 \\ 0 & \text{if } v \in V^{lb}_0 \\ \overline{p}^{lb,\min}_{v}(F) & \text{if } v \in V_{done} \setminus (V^{lb}_1 \cup V^{lb}_0) \\ \overline{p}^{lb,\min}_{v^{old}}(F) & \text{otherwise} \end{cases}$$

  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 3.2 and construction respectively, we have that:

$$\overline{p}^{lb,\min}_{v^{old}}(F) \leqslant \overline{p}^{lb,\min}_{v}(F) \; (= u^{lb}(v)) \tag{3}$$

$$\overline{p}^{lb,\min}_{v^{old}}(F) \leqslant x^{lb}_{init}(v)\,. \tag{4}$$

  Therefore, it remains to show that $x^{lb}_{init} \sqsubseteq F^{lb}_{V_{done}}(x^{lb}_{init})$. Now consider any $v \in V_1$, if $v \in V^1_{lb} \cup V^0_{lb} \cup V_{done}$, then by definition of $F^{lb}_{V_{done}}$ we have $F^{lb}_{V_{done}}(x^{lb}_{init})(v) = \overline{p}^{lb,\min}_{v}(F)$, and hence using (3) we have $F^{lb}_{V_{done}}(x^{lb}_{init})(v) \geqslant x^{lb}_{init}(v)$. On the other hand, if $v \notin V^1_{lb} \cup V^0_{lb} \cup V_{done}$, then by definition of $F^{lb}_{V_{done}}$:

$$\begin{aligned}
F^{lb}_{V_{done}}(x^{lb}_{init})(v) &= \min_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_{\mathrm{p}})(v_1) \cdot x^{lb}_{init}(v_1) \\
&\geqslant \min_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_{\mathrm{p}})(v_1) \cdot \overline{p}^{lb,\min}_{v^{old}_1}(F) && \text{by (4)} \\
&= \min_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v^{old}_1 \in V^{old}_1} \sum_{v_1 \subseteq v^{old}_1} \delta(v_{\mathrm{p}})(v_1) \cdot \overline{p}^{lb,\min}_{v^{old}_1}(F) && \text{since } P \preceq P_{old} \\
&= \min_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v^{old}_1 \in V^{old}_1} \delta_{old}(v_{\mathrm{p}})(v) \cdot \overline{p}^{lb,\min}_{v^{old}_1}(F) && \text{by Definition 7} \\
&= \min_{v^{old}_2 \leftarrow v^{old}} \min_{{v_{\mathrm{p}}}^{old} \leftarrow v^{old}_2} \sum_{v^{old}_1 \in V^{old}_1} \delta_{old}({v_{\mathrm{p}}}^{old})(v^{old}_1) \cdot \overline{p}^{lb,\min}_{v^{old}_1}(F) && \text{since } P \preceq P_{old} \\
&= \overline{p}^{lb,\min}_{v^{old}}(F) && \text{by Theorem 4.5} \\
&= x^{lb}_{init}(v) && \text{by construction.}
\end{aligned}$$

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

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

$$x^{ub}_{init}(v) = \begin{cases} 1 & \text{if } v \in V^{ub}_1 \\ 0 & \text{if } v \in V^{ub}_0 \\ \overline{p}^{ub,\min}_v(F) & \text{if } v \in V_{done} \setminus (V^{ub}_1 \cup V^{ub}_0) \\ \overline{p}^{lb,\min}_v(F) & \text{otherwise.} \end{cases}$$

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

$$\overline{p}^{lb,\min}_v(F) \leqslant \overline{p}^{ub,\min}_v(F) \; (= u^{ub}(v)) \tag{5}$$
$$\overline{p}^{lb,\min}_v(F) \leqslant x^{lb}_{init}(v)\,. \tag{6}$$

Therefore, it remains to show that $x^{ub}_{init} \sqsubseteq F^{ub}_{V_{done}}(x^{ub}_{init})$. Now consider any $v \in V_1$, if $v \in V^1_{lb} \cup V^1_{lb} \cup V_{done}$ then by construction $F^{ub}_{V_{done}}(x^{ub}_{init})(v) = \overline{p}^{ub,\min}_v(F)$, and hence using (5) we have $F^{ub}_{V_{done}}(x^{ub}_{init})(v) \geqslant x^{ub}_{init}(v)$. On the other hand, if $v \notin V^1_{lb} \cup V^1_{lb} \cup V_{done}$ then by definition of $F^{ub}_{V_{done}}$:

$$\begin{aligned} F^{ub}_{V_{done}}(x^{ub}_{init})(v) &= \max_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_{\mathrm{p}})(v_1) \cdot x^{ub}_{init}(v_1) \\ &\geqslant \max_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_{\mathrm{p}})(v_1) \cdot \overline{p}^{lb,\min}_{v_1}(F) && \text{by (6)} \\ &\geqslant \min_{v_2 \leftarrow v} \min_{v_{\mathrm{p}} \leftarrow v_2} \sum_{v_1 \in V_1} \delta(v_{\mathrm{p}})(v_1) \cdot \overline{p}^{lb,\min}_{v_1}(F) && \text{rearranging} \\ &= \overline{p}^{lb,\min}_v(F) && \text{by Theorem 4.7} \\ &= x^{ub}_{init}(v) && \text{by construction.} \end{aligned}$$

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

This completes the proof of (2), and hence that the value iteration scheme of Figure 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 [32]. 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");

[1] Supporting files for the models and properties used in the experimental results are available from `http://www.prismmodelchecker.org/subm/fmsd08/`.

- 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 [27]. 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 $\varepsilon{=}10^{-4}$. 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 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 that 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 [37] (4 refinement steps and a number of complex proofs) and the digital clocks semantics of [27] yields an abstract model with 1,212 states for $d{=}30$. For the Zeroconf example, the manual abstraction of [26] 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.

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 a result of the way that it splits elements of the partition 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.

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

Table 2: Performance statistics for the abstraction-refinement implementation

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, our implementation currently 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"). The overall performance of the framework is very encouraging and, longer term, we plan to replace the prototype abstraction tool with recently developed language-level MDP abstraction techniques, such as predicate abstraction [38, 24].

Table 3 also gives the time-savings obtained by using the optimisations described in Section 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

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

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

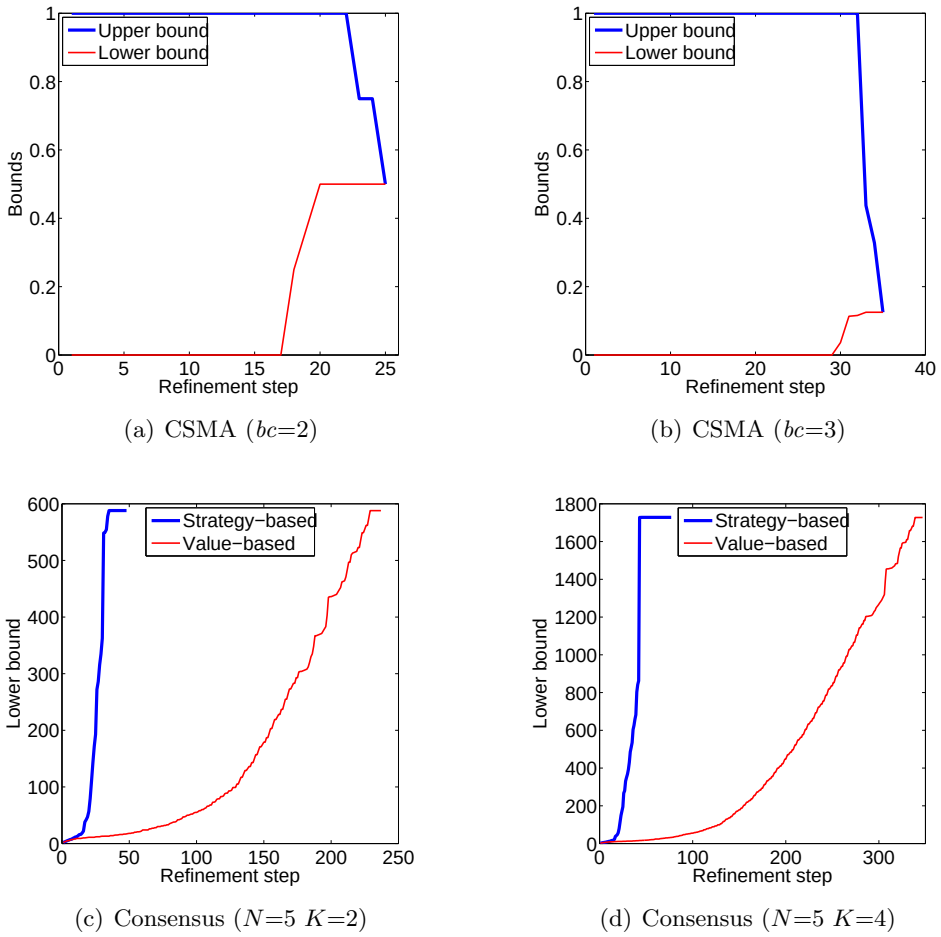
the fastest available engine in the tool). We see that, although the overall times for abstraction-refinement are slower than PRISM, this is again largely due to the fact that the (model-level) abstraction-construction component currently dominates the overall time requirements. On the whole, the abstraction-refinement framework performs very well.

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, Figure 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.

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 [23, 31]. Progress has been made in the area of qualitative probabilistic verification, see for example [39], and games have also been applied in the field of non-probabilistic model checking, for example [36]. Another approach to improving the efficiency of model checking for large Markov decision processes is through the use of partial order techniques [6, 17].

In [5], 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.



(a) CSMA (bc=2)

(b) CSMA (bc=3)

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

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

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

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 MLA results in [5] for Zeroconf (N=4, M=32), which has 26,121 concrete states, using the same accuracy of ε=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 Figure 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 [5] is not.

D'Argenio et al. [16] introduce an approach for verifying reachability properties of MDPs based on probabilistic simulation [33] 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). Because refinement is repeated until the probability (not the error bound) obtained falls below some threshold, a direct comparison is not feasible.

Although not employing refinement, we mention the tool PASS [38] which implements predicate abstraction on PRISM models. Since the abstraction is performed at the language level, it is applicable to infinite-state MDPs. However, like [16], this yields only one-sided bounds. Predicate abstraction of PRISM models using games was recently proposed in [24]; this work could potentially be used within our framework. Elsewhere, abstraction techniques have been considered for discrete-time Markov chains [19, 34], 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. [11] 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.

In [18] 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. Monniaux [30] also considers infinite state systems, demonstrating that the framework of abstract interpretation can be applied to Markov decision processes with infinite state spaces.

Finally, McIver and Morgan have developed a framework for the refinement and abstraction of probabilistic programs using expectation transformers [29]. The proof techniques developed in this work have been implemented in the HOL theorem-proving environment [22].

7 Conclusions

We have presented the first abstraction-refinement framework for game-based abstraction of Markov decision processes. Our novel abstraction technique is based on the translation of an MDP into a stochastic two-player game 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 are able to compute both lower and upper bounds on the minimum and maximum probability or expected reward of reaching a set of states. This provides valuable quantitative results with respect to both the behaviour of the original MDP and the utility of the abstraction, thus giving a basis for refining the abstraction.

Using this abstraction approach, we have introduced two refinement techniques and an optimised algorithm for the 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.

The next step in this research is to apply our framework at the level of MDP modelling formalisms (for example using the PRISM language or probabilistic versions of imperative languages such as C). Abstraction at the language level introduces the possibility of applying our technique to infinite-state MDPs. Research in this direction is currently underway, building on the techniques from [24].

References

[1] de Alfaro, L.: Formal verification of probabilistic systems. Ph.D. thesis, Stanford University (1997)

[2] de Alfaro, L.: Computing minimum and maximum reachability times in probabilistic systems. In: J. Baeten, S. Mauw (eds.) Proc. 10th Int. Conf. Concurrency Theory (CONCUR'99), *Lecture Notes in Computer Science*, vol. 1664, pp. 66–81. Springer (1999)

[3] de Alfaro, L., Henzinger, T., Kupferman, O.: Concurrent reachability games. In: Proc. 39th Symp. Foundations of Computer Science (FOCS'98), pp. 564–575. IEEE CS Press (1998)

[4] de Alfaro, L., Henzinger, T., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science **386**(3), 188–217 (2007)

[5] de Alfaro, L., Roy, P.: Magnifying-lens abstraction for Markov decision processes. In: W. Damm, H. Hermanns (eds.) Proc. 19th Int. Conf. Computer Aided Verification (CAV'07), *Lecture Notes in Computer Science*, vol. 4590, pp. 325–338. Springer (2007)

[6] Baier, C., Grosser, M., Ciesinski, F.: Partial order reduction for probabilistic systems. In: Proc. 1st Int. Conf.n Quantitative Evaluation of Systems (QEST'04), pp. 230–239. IEEE CS Press (2004)

[7] Baier, C., Kwiatkowska, M.: Model checking for a probabilistic branching time logic with fairness. Distributed Computing **11**(3), 125–155 (1998)

[8] Bertsekas, D., Tsitsiklis, J.: An analysis of stochastic shortest path problems. Mathematics of Operations Research **16**(3), 580–595 (1991)

[9] Billingsley, P.: Probability and Measure. John Wiley and Sons: New York (1979)

[10] Chatterjee, K., de Alfaro, L., Henzinger, T.: Trading memory for randomness. In: Proc. 1st Int. Conf. Quantitative Evaluation of Systems (QEST'04), pp. 206–217. IEEE CS Press (2004)

[11] Chatterjee, K., Henzinger, T., Jhala, R., Majumdar, R.: Counterexample-guided planning. In: Proc. 21st Conference in Uncertainty in Artificial Intelligence (UAI'05), pp. 104–111 (2005)

[12] Cheshire, S., Adoba, B., Guttman, E.: Dynamic configuration of IPv4 link-local addresses (draft August 2002). Zeroconf Working Group of the Internet Engineering Task Force (www.zeroconf.org)

[13] Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. In: A. Emerson, A. Sistla (eds.) Proc. 12th Int. Conf. Computer Aided Verification (CAV'00), *Lecture Notes in Computer Science*, vol. 1855, pp. 154–169. Springer (2000)

[14] Condon, A.: The complexity of stochastic games. Information and Computation **96**(2), 203–224 (1992)

[15] Condon, A.: On algorithms for simple stochastic games. Advances in computational complexity theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science **13**, 51–73 (1993)

[16] D'Argenio, P., Jeannet, B., Jensen, H., Larsen, K.: Reduction and refinement strategies for probabilistic analysis. In: H. Hermanns, R. Segala (eds.) Proc. 2nd Joint Int Workshop Process Algebra and Probabilistic Methods, Performance Modeling and Verification (PAPM/PROBMIV'01), *Lecture Notes in Computer Science*, vol. 2399, pp. 57–76. Springer (2001)

[17] D'Argenio, P., Niebert, P.: Partial order reduction on concurrent probabilistic programs. In: Proc. 1st Int. Conf. Quantitative Evaluation of Systems (QEST'04), pp. 240–249. IEEE CS Press (2004)

[18] Desharnais, J., Gupta, V., Jagadeesan, R., Panangaden, P.: Approximating labelled Markov processes. Information and Computation **184**(1), 160–200 (2003)

[19] Fecher, H., Leucker, M., Wolf, V.: Don't know in probabilistic systems. In: A. Valmari (ed.) Proc. 13th Int. SPIN Workshop on Model Checking of Software (SPIN'06), *Lecture Notes in Computer Science*, vol. 3925, pp. 71–88. Springer (2006)

[20] Graf, S., Saidi, H.: Construction of abstract state graphs with PVS. In: O. Grumberg (ed.) Proc. 9th Int. Conf. Computer Aided Verification (CAV'97), *Lecture Notes in Computer Science*, vol. 1254, pp. 72–83. Springer (1997)

[21] Hinton, A., Kwiatkowska, M., Norman, G., Parker, D.: PRISM: A tool for automatic verification of probabilistic systems. In: H. Hermanns, J. Palsberg (eds.) Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems

(TACAS'06), *Lecture Notes in Computer Science*, vol. 3920, pp. 441–444. Springer (2006)

[22] Hurd, J., McIver, A., Morgan, C.: Probabilistic guarded commands mechanized in HOL. Theoretical Computer Science **346**(1), 96–112 (2005)

[23] Huth, M.: An abstraction framework for mixed nondeterministic and probabilistic systems. In: C. Baier, B. Haverkort, H. Hermanns, J.P. Katoen, M. Siegle (eds.) Validation of Stochastic Systems, *Lecture Notes in Computer Science*, vol. 2925, pp. 419–444. Springer (2004)

[24] Kattenbelt, M., Kwiatkowska, M., Norman, G., Parker, D.: Game-based probabilistic predicate abstraction in PRISM. In: Proc. 6th Workshop Quantitative Aspects of Programming Languages (QAPL'08) (2008)

[25] Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. D. Van Nostrand Company (1966)

[26] Kwiatkowska, M., Norman, G., Parker, D.: Game-based abstraction for Markov decision processes. In: Proc. 3th Int. Conf. Quantitative Evaluation of Systems (QEST'06), pp. 157–166. IEEE CS Press (2006)

[27] Kwiatkowska, M., Norman, G., Parker, D., Sproston, J.: Performance analysis of probabilistic timed automata using digital clocks. Formal Methods in System Design **29**, 33–78 (2006)

[28] Larsen, K., Skou, A.: Bisimulation through probabilistic testing. Information and Computation **94**, 1–28 (1991)

[29] McIver, A., Morgan, C.: Abstraction, Refinement and Proof for Probabilistic Systems. Monographs in Computer Science. Springer (2004)

[30] Monniaux, D.: Abstract interpretation of programs as Markov decision processes. Science of Computer Programming **58**(1–2), 179 – 205 (2005)

[31] Norman, G.: Analyzing randomized distributed algorithms. In: C. Baier, B. Haverkort, H. Hermanns, J.P. Katoen, M. Siegle (eds.) Validation of Stochastic Systems, *Lecture Notes in Computer Science*, vol. 2925, pp. 384–418. Springer (2004)

[32] PRISM web site. http://www.prismmodelchecker.org/

[33] Segala, R.: Modelling and verification of randomized distributed real time systems. Ph.D. thesis, Massachusetts Institute of Technology (1995)

[34] Sen, K., Viswanathan, M., Agha, G.: Model-checking Markov chains in the presence of uncertainties. In: H. Hermanns, J. Palsberg (eds.) Proc. 12th Int. Conf. Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), *Lecture Notes in Computer Science*, vol. 3920, pp. 394–410. Springer (2006)

[35] Shapley, L.: Stochastic games. In: Proc. National Academy of Science, vol. 39, pp. 1095–1100 (1953)

[36] Shoham, S., Grumberg, O.: A game-based framework for CTL counter-examples and 3-valued abstraction-refinement. In: W. Hunt, F. Somenzi (eds.) Proc. 15th Int. Conf. Computer Aided Verification (CAV'03), *Lecture Notes in Computer Science*, vol. 2725, pp. 275–287. Springer (2003)

[37] Stoelinga, M., Vaandrager, F.: Root contention in IEEE 1394. In: J.P. Katoen (ed.) Proc. 5th Int. AMAST Workshop Real-Time and Probabilistic Systems (ARTS'99), *Lecture Notes in Computer Science*, vol. 1601, pp. 53–74. Springer (1999)

[38] Wachter, B., Zhang, L., Hermanns, H.: Probabilistic model checking modulo theories. In: Proc. 4th Int. Conf. Quantitative Evaluation of Systems (QEST'07), pp. 129–138. IEEE CS Press (2006)

[39] Zuck, L., Pnueli, A., Kesten, Y.: Automatic verification of probabilistic free choice. In: A. Cortesi (ed.) Proc. 3rd Int. Workshop Verification, Model Checking, and Abstract Interpretation (VMCAI'02), *Lecture Notes in Computer Science*, vol. 2294, pp. 208–224. Springer (2002)