Game-based Abstraction for Markov Decision Processes

Marta Kwiatkowska Gethin Norman David Parker
School of Computer Science, University of Birmingham
Birmingham, B15 2TT, UK
{mzk,gxn,dxp}@cs.bham.ac.uk

Abstract

In this paper we present a novel abstraction technique for Markov decision processes (MDPs), which are widely used for modelling systems that exhibit both probabilistic and nondeterministic behaviour. In the field of model checking, abstraction has proved an extremely successful tool to combat the state-space explosion problem. In the probabilistic setting, however, little practical progress has been made in this area. We propose an abstraction method for MDPs based on stochastic two-player games. The key idea behind this approach is to maintain a separation between nondeterminism present in the original MDP and nondeterminism introduced through abstraction, each type being represented by a different player in the game. Crucially, this allows us to obtain distinct lower and upper bounds for both the best and worst-case performance (minimum or maximum probabilities) of the MDP. We have implemented our techniques and illustrate their practical utility by applying them to a quantitative analysis of the Zeroconf dynamic network configuration protocol.

1. Introduction

Markov decision processes (MDPs) are a natural and widely used model for systems which exhibit both *nondeterminism*, due for example to concurrency, and *probability*, representing for example randomisation or unpredictability. Automatic verification of MDPs using probabilistic model checking has proved successful for analysing real-life systems from a wide range of application domains including communication protocols, security protocols, and randomised distributed algorithms. 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.

In this paper, we consider *abstraction* techniques, which have been established as one of the most effective ways of reducing the state-space explosion problem for non-probabilistic model checking (see e.g. [7]). The basic idea of such methods is to construct an abstract model, typically much smaller than the original (concrete) model, in which details not relevant to a particular property of interest have been removed. Such an abstraction is said to be *conservative* if satisfaction of a property in the abstract model implies that the property is also satisfied in the concrete model. For properties not satisfied in the abstract model, this is not the case, but information obtained during the verification process, such as a counterexample, maybe be used to refine the abstraction [6].

In the probabilistic setting, it is typically necessary to consider *quantitative* properties, in which case the actual probability of some behaviour being observed must be determined, e.g. "the probability of reaching an error state within T time units". Therefore in this setting a different notion of property preservation 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 must be considered. 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 since we are introducing additional uncertainty with regards to the precise behaviour of the system. The key idea in our abstraction approach is to maintain a distinction between the nondeterminism from the original MDP and the nondeterminism introduced during the abstraction process. To achieve this, we model abstractions as simple stochastic two-player games [8], where the two players correspond to the two different forms of nondeterminism. We can then analyse these models using techniques developed for such games [9, 14, 4].

Our analysis of these abstract models results in a separate lower and an upper bound for both the minimum and maximum probabilities (or expected reward) of reaching a set of states. This approach is particularly appealing since it also provides a quantitative measure of the utility of the abstraction. If the difference between the lower and upper bounds is too great, the abstraction can be refined and the process repeated. By comparison, if no discrimination between the two forms of nondeterminism is made, a single lower and upper bound would be obtained. In the (common) situation where the minimum and maximum probabilities (or expected rewards) are notably different, it is difficult to interpret the usefulness of the abstraction. 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 $[0, \varepsilon_1]$ and the maximum probability is in the interval $[1-\varepsilon_2, 1]$. 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.

Related Work. Below, we summarise work on abstraction methods for quantitative analysis of Markov decision processes and Markov chains. General issues relating to abstraction in the field of probabilistic model checking are discussed in [20, 28]. Progress has been made in the area of qualitative probabilistic verification, see for example [33], and games have also been applied in the field of non-probabilistic model checking, for example [32]. Another approach to improving the efficiency of model checking for large Markov decision processes is through the use of partial order techniques [3, 11].

D'Argenio et al. [10] introduce an approach for verifying quantitative reachability properties of MDPs based on probabilistic simulation [30]. Properties are analysed on abstractions obtained through successive refinements, starting from an initial coarse partition derived from the property under study. This approach only produces a lower bound for the minimum reachability probability and an upper bound for the maximum reachability probability and hence appears more suited to analysing Markov chains (models with discrete probabilities and no nondeterminism) since the minimum and maximum probabilities coincide in this case.

Huth [21] considers an abstraction approach for infinite state Markov chains where the abstract models (finite state approximations) contain probabilistic transitions labelled with intervals, rather than exact values. Conservative model checking of such models is achieved through a three-valued semantics of probabilistic computation tree logic (PCTL) [17]. Huth also proves the 'optimality' of the abstraction technique: for any finite set of until-free formulae, there always exists an abstraction in which satisfaction of each formula agrees with the concrete model. Fecher et al. [16] also consider an abstraction technique for Markov chains where probabilistic transitions are labelled with intervals and the logic PCTL has a three-valued interpretation. It is shown that model checking in this setting has the same complexity as that for standard Markov chains against PCTL. An alternative approach using Markov chains with intervals of probabilities can be found in [31].

In [15] 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 [27] 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 [26]. The proof techniques developed in this work have been implemented in the HOL theorem-proving environment [19].

Outline of the Paper. In the next section we present background material required for the remainder of the paper. In particular, we summarise results relating to Markov decision processes and to simple stochastic two player games. In Section 3 we describe our abstraction technique and show the correctness of the approach, then in Section 4 we illustrate its applicability through a case study concerning the Zeroconf dynamic network configuration protocol. Section 5 concludes the paper.

2. Background

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

2.1. 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* $\mathcal{M} = (S, s_{init}, Steps, rew)$*, where* S *is a set of states,* $s_{init} \in S$ *is the initial state,* $Steps : S \to 2^{\mathsf{Dist}(S)}$ *is the probability transition function and* $rew : S \times \mathsf{Dist}(S) \to \mathbb{R}_{\geq 0}$ *is the reward function.*

A probabilistic transition $s \xrightarrow{\mu} s'$ is made from a state s by first nondeterministically selecting a distribution $\mu \in Steps(s)$ and then making a probabilistic choice of target state s' according to the distribution μ. The reward function associates the non-negative real-value $rew(s,\mu)$ with performing the transition μ from the state s.

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:

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

such that $\mu_i(s_{i+1}){>}0$ for all i. We denote by $\pi(i)$ the $(i{+}1)$th state of π, $last(\pi)$ the last state of π if π is finite and $step(\pi,i)$ the distribution associated with the $(i{+}1)$-th transition (that is, $step(\pi,i) = \mu_i$). For any infinite path π and set of states F, the total reward accumulated until a state in F is reached along π, denoted $r(F,\pi)$, equals:

$$\textstyle\sum_{i=1}^{\min\{j \mid \pi(j)\in F\}} rew(\pi(i{-}1), step(\pi, i{-}1))$$

if there exists $j \in \mathbb{N}$ such that $\pi(j) \in F$, and equals ∞ otherwise. 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 is finite if and only if a state in F is reached from s with probability 1.

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 of $\mathcal{M}$ to a distribution $\mu \in Steps(last(\pi))$. For any state $s \in S$ and adversary A, let $Path^A_{fin}(s)$ and $Path^A(s)$ denote the sets of finite and infinite paths starting in s that correspond to A. Furthermore, let Adv denote the set of all adversaries.

Definition 2 *An adversary A is called* simple *(or memoryless) if for any finite paths π and π' for which $last(\pi) = last(\pi')$ we have $A(\pi) = A(\pi')$.*

The behaviour under a given adversary A is purely probabilistic and we can define a probability measure $Prob^A_s$ over the set of paths $Path^A(s)$ [22]. Below, we introduce two quantitative measures for MDPs which together form the basis for probabilistic model checking of MDPs [12, 1].

Probabilistic Reachability The first measure is *probabilistic reachability*, namely the minimum and maximum probability of reaching, from some state s, a set $F \subseteq S$ of target states. For a given adversary A, the probability of reaching F from s is given by:

$$p^A_s(F) \stackrel{\text{def}}{=} Prob^A_s\{\pi \in Path^A(s) \mid \exists i \in \mathbb{N} \,.\, \pi(i) \in F\} \,.$$

Definition 3 *For an MDP $\mathcal{M} = (S, s_{init}, Steps, rew)$, the* minimum and maximum reachability probabilities *of reaching the set of target states $F \subseteq S$ from a state $s \in S$ are defined as follows:*

$$p^{\min}_s(F) = \inf_{A\in Adv} p^A_s(F) \text{ and } p^{\max}_s(F) = \sup_{A\in Adv} p^A_s(F) \,.$$

Expected Reachability The second measure we consider is *expected reachability*, which refers to the expected reward accumulated, starting in state s, before reaching a set $F \subseteq S$ of target states. For an adversary $A \in Adv$, let $e^A_s(F)$ denote the usual expectation of the function $r(F,\cdot)$:

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

Definition 4 *For an MDP $\mathcal{M} = (S, s_{init}, Steps, rew)$, the* minimum and maximum expected rewards *of reaching a set of target states $F \subseteq S$ from the state $s \in S$ are defined as follows:*

$$e^{\min}_s(F) = \inf_{A\in Adv} e^A_s(F) \text{ and } e^{\max}_s(F) = \sup_{A\in Adv} e^A_s(F) \,.$$

Computing values for expected reachability (and probability) reduces to the *stochastic shortest path problem* for Markov decision processes; see for example [2, 13]. A key result in this respect is that optimality with respect to probabilistic and expected reachability can always be achieved with simple adversaries (see Definition 2). A consequence of this is that these quantities can be computed through an iterative processes known as *value iteration*, the basis of which is given in the lemma below.

Lemma 5 *Consider an MDP $\mathcal{M} = (S, s_{init}, Steps, rew)$ and set of target states F. Let F_0 be the set of states from which F cannot be reached. The following sequence of vectors converges to the minimum probability of reaching the target set F. Let $(p_n)_{n\in\mathbb{N}}$ be the sequence of vectors over S such that for any state $s \in S$, if $s \in F$ or $s \in F_0$ then $p_n(s) = 1$ and $p_n(s) = 0$ respectively for all $n \in \mathbb{N}$, and otherwise $p_0(s) = 0$ and for any $n \in \mathbb{N}$:*

$$p_{n+1}(s) = \min_{\mu\in steps(s)} \sum_{s'\in S} \mu(s') \cdot p_n(s')$$

The maximum probability and the minimum or maximum expected reward of reaching the target set F can be defined in a similar fashion [2, 13].

2.2. Simple Stochastic Games

In this section we review simple stochastic games [8], which are turn-based games involving two players and chance.

Definition 6 *A* turn-based stochastic game *is a tuple $\mathcal{G} = ((V,E), v_{init}, (V_1, V_2, V_\bigcirc), \delta, rew)$ where:*

- *(V,E) is a finite directed graph;*
- *$v_{init} \in V$ is the initial vertex;*
- *$(V_1, V_2, V_\bigcirc)$ is a partition of V;*
- *$\delta : V_\bigcirc \to \mathsf{Dist}(V)$ is the probabilistic transition function;*

- *$rew : E \to \mathbb{R}_{\geq 0}$ is the reward function over edges.*

Vertices in V_1, V_2 and $V_{\bigcirc}$ 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 $v \in V_{\bigcirc}$ then v' is selected randomly according to the distribution $\delta(v)$.

A Markov decision process can be thought of as a turn-based 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 $\mathcal{G}$ is a sequence $\omega = \langle v_0 v_1 v_2 \ldots \rangle$ such that $(v_i, v_{i+1}) \in E$ for all $i \in \mathbb{N}$. We denote by $\omega(i)$ the ith vertex in the play and by $last(\omega)$ the last vertex of ω if ω is finite. For any infinite play ω and set of vertices F, the total reward accumulated until a vertex in F is reached, denoted $r(F, \omega)$, equals:

$$\textstyle\sum_{i=1}^{\min\{j \,|\, \omega(j) \in F\}} rew(\omega(i-1), \omega(i))$$

if there exists $j \in \mathbb{N}$ such that $\omega(j) \in F$, and equals ∞ otherwise.

A *strategy* for player 1 is a function $\sigma_1 : V^* V_1 \to \mathsf{Dist}(V)$ such that for any $\omega \in V^* V_1$ and $v \in V$, if $\sigma_1(\omega)(v) > 0$, then $(last(\omega), v) \in E$. 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 a fixed strategy pair, the behaviour of the game is completely random and, for any vertex v, we can construct a probability measure $Prob_v^{\sigma_1,\sigma_2}$ over the set of infinite plays $Play^{\sigma_1,\sigma_2}(v)$. This construction proceeds similarly to MDPs [22].

Reachability Objectives A *reachability objective* is a set of vertices F which a player attempts to reach. For fixed strategies σ_1 and σ_2 and vertex $v \in V$ we define both the probability and expected reward corresponding to the reachability objective F as:

$$\begin{aligned} p_v^{\sigma_1,\sigma_2}(F) &\stackrel{\text{def}}{=} Prob_v^{\sigma_1,\sigma_2}\{\omega \,|\, \exists i \in \mathbb{N} \wedge \omega(i) \in F\} \\ e_v^{\sigma_1,\sigma_2}(F) &\stackrel{\text{def}}{=} \int_{\omega \in Play^{\sigma_1,\sigma_2}(v)} r(F,\omega)\, \mathrm{d}Prob_v^{\sigma_1,\sigma_2} . \end{aligned}$$

The optimal probabilities of the game for player 1 and player 2, with respect to the reachability objective F, are defined as follows:

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

and the optimal expected rewards are:

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

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

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

The optimal strategies for player 2 and for expected rewards can be defined analogously.

We now summarise results from [4, 8, 9].

Definition 7 *A strategy σ_i is* pure *if it does not use randomisation, that is, for any finite play ω such that $last(\omega) \in V_i$, there exists $v' \in V$ such that $\sigma_i(\omega)(v') = 1$. A strategy σ_i is* memoryless *if its choice depends only on the current vertex, that is, $\sigma_i(\omega) = \sigma_i(\omega')$ for any finite plays ω and ω' such that $last(\omega) = last(\omega')$.*

Proposition 8 *Let $\mathcal{G}$ be any simple stochastic game and F a set of target vertices. The family of pure memoryless strategies suffices for optimality with respect to reachability objectives.*

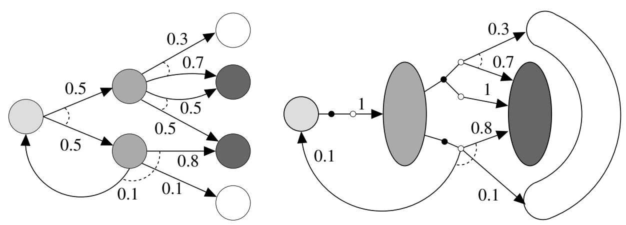
Lemma 9 *Consider a turn based stochastic game $\mathcal{G} = ((V, E), v_{init}, (V_1, V_2, V_{\bigcirc}), \delta, rew)$ and set of target vertices F. Let F_0 be the set of vertices from which F cannot be reached. The following sequence of vectors converges to the vector of optimal probabilities for player 1 with respect to the reachability objective F. Let $(p_n)_{n \in \mathbb{N}}$ be the sequence of vectors over V such that for any vertex $v \in V$, if $v \in F$ or $v \in F_0$ then $p_n(v) = 1$ and $p_n(v) = 0$ respectively for all $n \in \mathbb{N}$, and otherwise $p_0(v) = 0$ and for any $n \in \mathbb{N}$:*

- *$p_{i+1}(v) = \max_{(v,v') \in E} p_i(v')$ if $v \in V_1$;*
- *$p_{i+1}(v) = \min_{(v,v') \in E} p_i(v')$ if $v \in V_2$;*
- *$p_{i+1}(v) = \sum_{v' \in V} \delta(v)(v') \cdot p_i(v')$ if $v \in V_{\bigcirc}$.*

Lemma 9 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 and for expected rewards. Observe the similarity between this and the value iteration method for MDP solution described in Section 2.1.

3. Abstraction for MDPs

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 two-player stochastic game where the choices made by one player (player 2) correspond to the nondeterminism in the original MDP and the choices made by the other (player 1) correspond to the nondeterminism introduced by the abstraction process. The abstract MDP is defined by a partition $P_S = \{S_1, S_2, \ldots, S_n\}$ of the state space S, which we assume to be provided by the user. In practice, this might for example be derived, as in predicate abstraction, via the definition of a set of predicates on the



(a) Original MDP (b) Abstraction (simple game)

Figure 1. Abstraction of a simple MDP

variables of the concrete state space. In this paper, we do not consider the issue of finding an appropriate partition.

In the following, for any distribution μ over S, we denote by $\overline{\mu}$ the probability distribution over P_S lifted from μ, i.e. $\overline{\mu}(S_i) = \sum_{s \in S_i} \mu(s)$ for all $S_i \in P_S$.

Definition 10 *Given an MDP $\mathcal{M} = (S, s_{init}, Steps, rew)$ and a partition of the state space P_S we define the corresponding* abstract MDP *as the turn-based stochastic game*

$$\mathcal{G}_{\mathcal{M},P_S} = ((V, E), v_{init}, (V_1, V_2, V_{\bigcirc}), \delta, \overline{rew})$$

in which:

- $V_1 = P_S$*;*
- $V_2 \subseteq 2^{\mathsf{Dist}(P_s)}$ *where $v \in V_2$ if and only if there exists $s \in S$ such that $v = \{\overline{\mu} \mid \mu \in Steps(s)\}$;*
- $V_{\bigcirc} = \{\overline{\mu} \mid \mu \in Steps(s)$ *for some* $s \in S\}$*;*
- $v_{init} = S_i$ *where $s_{init} \in S_i$;*
- $(v, v') \in E$ *if and only if one of the following conditions holds:*
  - $v \in V_1$*,* $v' \in V_2$ *and* $v' = \{\overline{\mu} \mid \mu \in Steps(s)\}$ *for some* $s \in v$*;*
  - $v \in V_2$*,* $v' \in V_{\bigcirc}$ *and* $v' \in v$*;*
  - $v \in V_{\bigcirc}$*,* $v' \in V_1$ *and* $v(v'){>}0$*;*
- $\delta : V_{\bigcirc} \to \mathsf{Dist}(V)$ *is the identity function;*
- $\overline{rew}(v, v')$ *equals* $rew(s, \mu)$ *if* $(v, v') \in V_2 \times V_{\bigcirc}$ *and there exists $s \in S$ and $\mu \in Steps(s)$ such that $v = \{\overline{\mu} \mid \mu \in Steps(s)\}$ and $v' = \overline{\mu}$, and equals 0 otherwise.*

Note that for the reward function $\overline{rew}$ to be well defined we require that for any $s, s' \in S$, if $\{\overline{\mu} \mid \mu \in Steps(s)\} = \{\overline{\mu} \mid \mu \in Steps(s')\}$, then $rew(s, \mu) = rew(s', \mu')$ for all $\mu, \mu' \in \mathsf{Dist}(S)$ such that $\overline{\mu} = \overline{\mu}'$. We can consider this restriction as saying that the abstraction can only be applied if the reward function rew is compatible with the state partition P_S.

Example 11 *We illustrate the abstraction process on a simple example, shown in Figure 1(a), where the state partition of the MDP is indicated by the different shadings of the states. The abstract MDP is given in Figure 1(b): the large, shaded shapes are player 1 vertices (V_1), player 2 vertices (V_2) are denoted by small black circles, and probabilistic vertices ($V_{\bigcirc}$) by small white circles.*

Intuitively, the roles of the vertices and players in the abstract MDP can be understood as follows. A V_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 11 (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 vertices in V_2 are actually sets of probability distributions, not 'concrete' states. Hence, all states with the same outgoing distributions are collapsed onto one. In fact there is a greater reduction since it is those states with the same outgoing distributions defined over the abstract states that are collapsed. 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 $V_{\bigcirc}$ vertices, we need only store the vertices in V_1, the outgoing transitions comprising each probability distribution from V_1, and how these transitions are grouped (into elements of V_2 and into individual probability distributions). Later, in Section 4 we will show how, on a complex case study, the abstraction process brings a significant reduction in model size.

3.1. Analysis of the Abstract MDP

We now describe how, from the abstract MDP $\mathcal{G}_{\mathcal{M},P_S}$ for an MDP $\mathcal{M}$ and state partition P_S, we derive lower and upper bounds for probabilistic reachability and expected reachability properties, namely bounds for the values $p_s^{\min}(F)$, $p_s^{\max}(F)$, $e_s^{\min}(F)$ and $e_s^{\max}(F)$ for a set of target states $F \subseteq S$. We assume, without loss of generality, that the set F is an element of P_S. We assume also that the state partition preserves the value of the reward function of $\mathcal{M}$ as described in Section 3.

Theorem 12 *Let $\mathcal{M} = (S, s_{init}, Steps, rew)$ be an MDP and $P_S = \{S_1, \ldots, S_n\}$ a partition of the state space S. For a set of target states $F \in P_S$, consider the simple stochastic game $\mathcal{G}_{\mathcal{M},P_S} = ((V, E), v_{init}, (V_1, V_2, V_{\bigcirc}), \delta, \overline{rew})$ constructed according to Definition 10. Then, for any state*

$s \in S$:

$$\begin{array}{rcccl}
\inf_{\sigma_1,\sigma_2} p_v^{\sigma_1,\sigma_2}(F) & \le & p_s^{\min}(F) & \le & \sup_{\sigma_1}\inf_{\sigma_2} p_v^{\sigma_1,\sigma_2}(F) \\
\sup_{\sigma_2}\inf_{\sigma_1} p_v^{\sigma_1,\sigma_2}(F) & \le & p_s^{\max}(F) & \le & \sup_{\sigma_1,\sigma_2} p_v^{\sigma_1,\sigma_2}(F)
\end{array}$$

and

$$\begin{array}{rcccl}
\inf_{\sigma_1,\sigma_2} e_v^{\sigma_1,\sigma_2}(F) & \le & e_s^{\min}(F) & \le & \sup_{\sigma_1}\inf_{\sigma_2} e_v^{\sigma_1,\sigma_2}(F) \\
\sup_{\sigma_2}\inf_{\sigma_1} e_v^{\sigma_1,\sigma_2}(F) & \le & e_s^{\max}(F) & \le & \sup_{\sigma_1,\sigma_2} e_v^{\sigma_1,\sigma_2}(F)
\end{array}$$

where v is the unique vertex of V_1 such that $s \in v$.

Proof Outline. The proof relies on Proposition 8 and constructing, for any adversary A of the MDP and state s, strategies σ_1 and σ_2 for player 1 and player 2, respectively, such that $p_v^{\sigma_1,\sigma_2}(F) = p_s^A(F)$ and $e_v^{\sigma_1,\sigma_2}(F) = e_s^A(F)$. The full proof can be found in [24]. □

We can iteratively compute values for the bounds in the above theorem that correspond to the optimal values for either player 1 or player 2 (see Lemma 9). The remaining bounds, although not usually considered in the context of two player games (because the two players cooperate) can be computed by considering the game as an MDP and applying conventional value iteration (see Lemma 5).

Example 13 *Let us return to the previous example (see Example 11 and Figure 1). Suppose that we are interested in the probability in the original MDP of, starting from the leftmost state, reaching the darkly shaded states on the right hand side. 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 12, 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.*

On the other hand, if 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 $[0.7, 1]$.

4. Case Study

We now demonstrate the applicability of our approach to a case study: the Zeroconf protocol [5] 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 protocol functions as follows. A new device, or *host*, wishing to join such a network randomly selects an IP address from a set of 65,024 addresses allocated for this purpose. It then broadcasts several messages, called *probes*, to the other hosts in the network in an attempt to determine if this address is already in use. The probes operate as requests to use the address and contain the IP address selected by the host. Hosts in the network already operating with IP addresses different to that chosen by the new host ignore these probes. If, however, a host receives a probe containing the IP address that it is currently using, it responds with an *ARP packet*, asserting its claim to the address. If the new host receives a reply (an ARP packet) to a probe it has sent, then it reconfigures (randomly selects an IP address and starts sending probes with this new address). If after sending four such probes no reply is received within a certain time bound, the host begins to use the address.

The Zeroconf protocol specifies precisely the timing of the various messages, for example, that the four probes are to be sent at two second intervals. Clearly, though, the precise latency of these messages over the network is unknown. Furthermore, it assumed that there is a certain probability that messages can be lost during transmission. Hence, it is possible that the new device will end up using an IP address that is already in use.

4.1. The Model

We use a slightly simplified version of the model of Zeroconf from [25]. We model the situation where a new device joins a network of N existing hosts, in which there are a total of M IP addresses available. We assume that the communication medium between the new host and each existing host in the network is such that messages arrive in the order in which they are sent. We suppose that there are a variety of message propagation delays between the new host and the existing hosts (for example the propagation delay for one host is between 0.1 and 0.2 seconds while for another it is between 0.6 and 0.9) and that message loss probabilities are proportional to these delays. For the full model, see [24].

The concrete (full) model contains $2N+1$ components: the new host and N pairs of channels for the two-way communication between the new host and each of the configured hosts. Since the other hosts do nothing except ignore or reply to messages, they are not modelled explicitly. The state of the new host comprises its program counter, the IP address it has currently selected, a count of how many probes it has sent and a clock to measure time between probes. The state of each of the $2N$ channels comprises its status (whether it is empty, has a message to send or is sending a message), the IP address (if any) which is currently being transmitted and a clock measuring the time elapsed since the message was sent.

The partition of the state space of the concrete model, which defines the abstract model, is in this case given by a mapping to a reduced set of variables. For the new host, we

N	M=32	M=64	abstraction
4	26,121 (50,624)	50,377 (98,080)	737 (1,594)
5	58,497 (139,104)	113,217 (270,272)	785 (1,678)
6	145,801 (432,944)	282,185 (839,824)	833 (1,762)
7	220,513 (614,976)	426,529 (1,189,792)	857 (1,806)
8	432,185 (1,254,480)	838,905 (2,439,600)	881 (1,850)

Table 1. Model statistics: states (transitions)

replace the IP addresses (range $1, \ldots, M$) with two values 1 and 2 (in both models the value 0 is also used to indicate that the host currently has no IP address selected), where 1 represents the set of fresh IP addresses and 2 denotes those already in use (the IP addresses of the other hosts).

For the channels, we partition the local states according to which of the following condition is satisfied:

- no messages are being sent;
- a broadcast initiated by the new host is in progress, x time units have elapsed since the broadcast began, n messages have still not arrived and the type of IP address in the message, ip, is 1 or 2 (a fresh IP address or an IP address of one of the configured hosts);
- a configured host is sending an ARP packet to the new host, x time units have elapsed, and the type of the IP address in the packet is ip (as in the previous case).

We implemented a prototype Java implementation of the MDP abstraction process described in Section 3 and then applied it to a range of concrete models of the Zeroconf protocol constructed with PRISM [18, 29]. The sizes of the resulting models (number of states and transitions) can be seen in Table 1 (recall that N denotes the number of hosts with configured IP address and M denotes the number of available IP addresses). As the results demonstrate, the abstraction provides a very significant reduction in model size, both in terms 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. This fact can be understood from the description of the abstraction process above: in the abstract model we only keep track of the number of configured hosts that have yet to receive a broadcast from the abstract host and store only 2 “abstract” IP addresses (representing the set of fresh IP addresses and the set of addresses already in use).

We note that the current limitation in the size of models we have considered (values of M and N) is due to fact that we have chosen to present results for both the concrete and abstract models. However, as Table 1 indicates, in the case of the abstract model it will be possible to verify the models generated for much larger values of both M and N.

M	N	lower bound	actual value	upper bound
32	4	0.99993760	0.99997866	0.99999984
	5	0.99991920	0.99997575	0.99999976
	6	0.99989917	0.99997248	0.99999971
	7	0.99987739	0.99997097	0.99999967
	8	0.99985420	0.99996896	0.99999949
64	4	0.99997099	0.99999023	0.99999993
	5	0.99996310	0.99998894	0.99999991
	6	0.99995489	0.99998776	0.99999990
	7	0.99994652	0.99998751	0.99999987
	8	0.99993776	0.99998722	0.99999984

Table 2. Minimum probability that the new host eventually succeeds in selecting a fresh IP address

4.2. Experimental Results

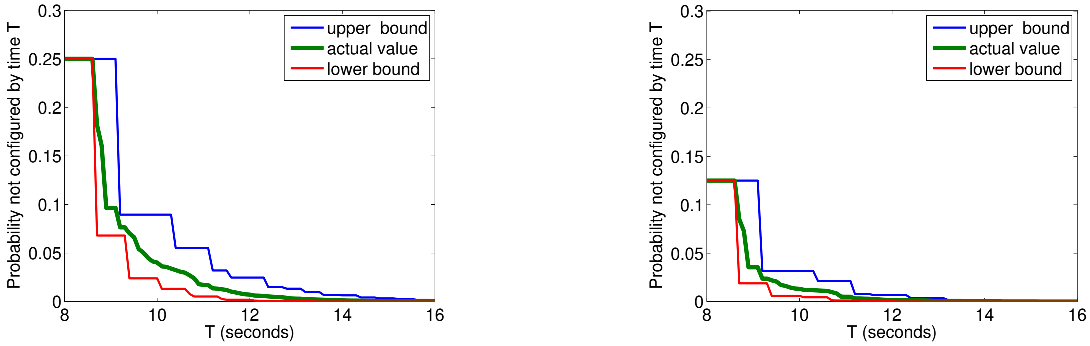
To validate the abstracted model we studied three properties of the Zeroconf model:

- the probability that the new host eventually succeeds in configuring an IP address not already in use;
- the probability that the new host succeeds in configuring an IP address not already in use within a fixed time bound;
- the expected time for the new host to complete the protocol (start using an IP address).

For each, we consider the best- and worst-case (i.e. minimum or maximum values). To allow for the computation of the expected time properties we have defined the reward function of the concrete model such that the reward of transitions corresponding to letting time pass equals the time that elapses when this transition is performed and all other transitions have reward 0.

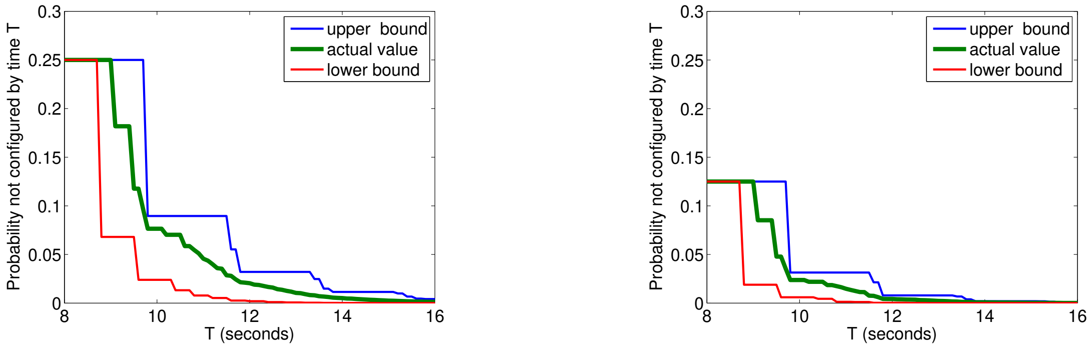
We apply model checking both to the concrete models (to establish the exact minimum and maximum values) and to the abstract models (to compute lower and upper bounds on these values). The former is done in conventional manner with PRISM; the latter is done with our prototype implementation of the algorithm described in Section 3.

Table 2 shows the minimum probability of eventually selecting an unused IP address (the maximum probability is the same in this case). Figure 2 and Figure 3 presents results concerning the minimum and maximum probability that the new host succeeds in configuring an IP address not already in use within a fixed time bound T. We have used the same vertical scale in Figure 2 and Figure 3 in order to allow a comparison of the results obtained for the different values of M and N and when considering either minimum or maximum probabilities. Finally, in Table 3 we give results ob-



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

Figure 2. Minimum probability that the new host configures successfully by time T



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

Figure 3. Maximum probability that the new host configures successfully by time T

tained for the minimum and maximum expected time for a host to complete the protocol (start using an IP address).

For all of the values of M and N considered we see that there is relatively little difference between the minimum and maximum cases. In fact, the minimum and maximum probability that the new host eventually successfully selects a fresh IP address actually coincide. These probabilities are the same because the only nondeterminism in the original system relates to the timing characteristics of the protocol. For the remaining properties, the similarity between the minimum and maximum cases (as shown in Figures 2 and 3 and Table 3) is due to the fact that there is actually only a small probability of a situation arising where the new host picks an IP address that is in use by one of the other hosts, and hence that the remaining hosts have any effect on the behaviour of the configuring host. Another interesting characteristics of the graphs in Figures 2 and 3 is that the plots are not smooth. This is a consequence of the discrete nature of the protocol: the new host waits for 2 seconds between sending probes and sends 4 probes before it starts using an IP address.

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 and Figure 3, 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 results presented in this section, it can be seen that our abstraction approach results in tight approximations for the performance characteristics of the protocol while at the same time producing a significant

M	N	minimum expected time			maximum expected time		
		lower bound	actual value	upper bound	lower bound	actual value	upper bound
32	4	8.108767	8.157220	8.219029	8.123052	8.246489	8.304700
	5	8.140993	8.203499	8.283920	8.159511	8.318295	8.394968
	6	8.175700	8.253337	8.353800	8.198776	8.395620	8.492185
	7	8.213181	8.293923	8.429265	8.241181	8.457891	8.597169
	8	8.253788	8.337890	8.511020	8.287120	8.525360	8.710893
64	4	8.050758	8.073371	8.102218	8.057425	8.115030	8.142199
	5	8.064523	8.093128	8.129936	8.072997	8.145661	8.180759
	6	8.078762	8.113567	8.158610	8.089107	8.177352	8.220647
	7	8.093501	8.128917	8.188289	8.105781	8.200838	8.261937
	8	8.108767	8.144815	8.219029	8.123052	8.225162	8.304701

Table 3. Minimum and maximum expected time for completion of the protocol

reduction in the state space.

Comparing the results for the different values of N and M, we find that for all three properties of the model that we have considered, the differences in bounds obtained become smaller with either an increase in M or a decrease in N. This is due to the fact that either incrementing M or decrementing N increases the probability of the new host choosing a fresh IP address, and therefore reduces the probability of the remaining hosts influencing its behaviour.

5. Conclusions

We have presented a novel approach for applying abstraction to the probabilistic verification of Markov decision processes. Our technique is based on the translation of an MDP into a (usually significantly smaller) 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 applied. Our prototype implementation has allowed us to demonstrate the potential of this approach on a complex case study.

We hope to extend this work in a number of directions. Firstly, we are aiming to adapt the abstraction process so that it can be applied at the level of the modelling formalism used (in this case the PRISM language). This would allow us to bypass the construction of the full MDP which could otherwise be a problem when considering very large models. In addition, performing the abstraction at the language level introduces the possibility of applying our technique to infinite state MDPs. We anticipate that, due to the similarity of the numerical methods (Lemma 5 and Lemma 9), the symbolic methods developed in PRISM [23] can be extended to solving simple stochastic games constructed in the abstraction process. We also intend to look at ways of automatically or semi-automatically generating partitions based on the predicates appearing in both the specification of the model and those appearing in the properties of interest.

Acknowledgements

The authors are supported in part by EPSRC grants GR/S11107 and GR/S46727 and Microsoft Research Cambridge contract MRL 2005-44.

References

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

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

[3] C.Baier, M. Grosser, and F. Ciesinski. Partial order reduction for probabilistic systems. In *Proc. QEST'04*, pages 230–239. IEEE Computer Society Press, 2004.

[4] K. Chatterjee, L. de Alfaro, and T. Henzinger. Trading memory for randomness. In *Proc. QEST'04*, pages 206–217. IEEE Computer Society Press, 2004.

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

[6] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In A. Emerson and A. Sistla, editors, *Proc. CAV'00*, volume 1855 of *LNCS*, pages 154–169. Springer, 2000.

[7] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. The MIT Press, 1999.

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

[9] A. Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory, DIMACS Series*

in Discrete Mathematics and Theoretical Computer Science, 13:51–73, 1993.

[10] P. D'Argenio, B. Jeannet, H. Jensen, and K. Larsen. Reachability analysis of probabilistic systems by successive refinements. In L. de Alfaro and S. Gilmore, editors, *Proc. PAPM/PROBMIV'01*, volume 2165 of *LNCS*, pages 39–56. Springer, 2001.

[11] P. D'Argenio and P. Niebert. Partial order reduction on concurrent probabilistic programs. In *Proc. QEST'04*, pages 240–249. IEEE Computer Society Press, 2004.

[12] L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, 1997.

[13] L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J. Baeten and S. Mauw, editors, *Proc. CONCUR'99*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.

[14] L. de Alfaro, T. Henzinger, and O. Kupferman. Concurrent reachability games. In *Proc. FOCS'98*, pages 564–575. IEEE Computer Society Press, 1998.

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

[16] H. Fecher, M. Leucker, and V. Wolf. Don't know in probabilistic systems. In A. Valmari, editor, *Proc. SPIN'06*, volume 3925 of *LNCS*, pages 71–88. Springer, 2006.

[17] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[18] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, editors, *Proc. TACAS'06*, volume 3920 of *LNCS*, pages 441–444. Springer, 2006.

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

[20] H. Huth. An abstraction framework for mixed nondeterministic and probabilistic systems. In C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of *LNCS*, pages 419–444. Springer, 2004.

[21] M. Huth. On finite-state approximants for probabilistic computation tree logic. *Theoretical Computer Science*, 346(1):113–134, 2005.

[22] J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains*. D. Van Nostrand Company, 1966.

[23] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2):128–142, 2004.

[24] M. Kwiatkowska, G. Norman, and D. Parker. Game-based abstraction for Markov decision processes. Technical Report CSR-06-05, School of Computer Science, University of Birmingham, 2006.

[25] M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. In K. Larsen and P. Niebert, editors, *Proc. FORMATS'03*, volume 2791 of *LNCS*, pages 105–120. Springer, 2003.

[26] A. McIver and C. Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems*. Monographs in Computer Science. Springer, 2004.

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

[28] G. Norman. Analyzing randomized distributed algorithms. In C. Baier, B. Haverkort, H. Hermanns, J.-P. Katoen, and M. Siegle, editors, *Validation of Stochastic Systems: A Guide to Current Research*, volume 2925 of *LNCS*, pages 384–418. Springer, 2004.

[29] PRISM web site. www.cs.bham.ac.uk/˜dxp/prism.

[30] R. Segala. *Modelling and Verification of Randomized Distributed Real Time Systems*. PhD thesis, Massachusetts Institute of Technology, 1995.

[31] K. Sen, M. Viswanathan, and G. Agha. Model-checking Markov chains in the presence of uncertainties. In H. Hermanns and J. Palsberg, editors, *Proc. TACAS'06*, volume 3920 of *LNCS*, pages 394–410. Springer, 2006.

[32] S. Shoham and O. Grumberg. A game-based framework for CTL counter-examples and 3-valued abstraction-refinement. In W. Hunt and F. Somenzi, editors, *Proc. CAV'03*, volume 2725 of *LNCS*, pages 275–287. Springer, 2003.

[33] L. Zuck, A. Pnueli, and Y. Kesten. Automatic verification of probabilistic free choice. In A. Cortesi, editor, *Proc. VMCAI'02*, volume 2294 of *LNCS*, pages 208–224. Springer, 2002.