Motion Planning under Partial Observability using Game-Based Abstraction

Leonore Winterer[1], Sebastian Junges[2], Ralf Wimmer[1], Nils Jansen[3],
Ufuk Topcu[3], Joost-Pieter Katoen[2], and Bernd Becker[1]

***Abstract*— We study motion planning problems where agents move inside environments that are not fully observable and subject to uncertainties. The goal is to compute a strategy for an agent that is guaranteed to satisfy certain safety and performance specifications. Such problems are naturally modeled by partially observable Markov decision processes (POMDPs). Because of the potentially huge or even infinite belief space of POMDPs, verification and strategy synthesis is in general computationally intractable. We tackle this difficulty by exploiting typical structural properties of such scenarios; for instance, we assume that agents have the ability to observe their own positions inside an environment. Ambiguity in the state of the environment is abstracted into non-deterministic choices over the possible states of the environment. Technically, this abstraction transforms POMDPs into probabilistic two-player games (PGs). For these PGs, efficient verification tools are able to determine strategies that approximate certain measures on the POMDP. If an approximation is too coarse to provide guarantees, an abstraction refinement scheme further resolves the belief space of the POMDP. We demonstrate that our method improves the state of the art by orders of magnitude compared to a direct solution of the POMDP.**

I. Introduction

Offline motion planning for dynamical systems with uncertainties aims at finding a *strategy* for an agent that ensures certain desired behavior [1]. Planning scenarios that exhibit stochastic behavior are naturally modeled by Markov decision processes (MDPs). An MDP is a non-deterministic model in which the agent chooses to perform an action under full knowledge of the environment it is operating in. The outcome of the action is a distribution over the states. For many robotic applications, however, information about the current state of the environment is not *observable* [2]–[4]. In such scenarios, where the actual state of the environment is not exactly known, the model is a *partially observable Markov decision process* (POMDP). By tracking the observations made while it moves, an agent can infer the likelihood of the environment being in a certain state. This likelihood is called the *belief state* of the agent. Executing an action leads to an update of the belief state because new observations are made. The belief state together with an update function form a (possibly infinite) MDP, commonly referred to as the underlying *belief MDP* [5].

As an example, take a scenario where a controllable agent needs to traverse a room while avoiding static obstacles and randomly moving opponents whose positions cannot always be observed by the agent. The goal is to determine a strategy for the agent, that (provably) ensures safe traversal with a certain high probability.

[1]Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany
[2]RWTH Aachen University, Aachen, Germany
[3]The University of Texas at Austin, Austin, TX, USA

Quantitative verification techniques like *probabilistic model checking* [6] provide comprehensive guarantees on such a strategy. For finite MDPs, tools like `PRISM` [7] or `storm` [8] employ efficient model checking algorithms to asses the probability to reach a certain set of states. However, POMDP verification suffers from the large, potentially infinite belief space, and is intractable even for rather small instances.

Approach: We outline the approach and the structure of the paper in Fig. 1, details in the figure will be discussed in the respective sections. Starting from a *problem description*, we propose to use an *encoding* of the problem as a POMDP. We observe that motion planning scenarios as described above naturally induce certain structural properties in the POMDP. In particular, we assume that the agent can observe its own position while the exact position of the opponents is observable only if they are nearby according to a given distance metric. We propose an *abstraction method* that, intuitively, lumps states inducing the same observations. Since it is not exactly known in which state of the environment a certain action is executed, a non-deterministic choice over these lumped states is introduced. Resolving this choice induces a new level of non-determinism into the system in addition to the choices of the agent: The POMDP abstraction results in a *probabilistic two-player game* (PG) [10]. The agent is the first player choosing an action while the second player chooses in which of the possible (concrete) states the action is executed. Model checking computes an *optimal strategy* for the agent on this PG.

The *automated* abstraction procedure is inspired by *game-based abstraction* [10], [11] of potentially infinite MDPs, where states are lumped in a similar fashion. We show that our approach is *sound* in the sense that a strategy for the agent in the PG defines a strategy for the original POMDP. Guarantees for the strategy carry over to POMDPs, as it induces bounds on probabilities. As we target an undecidable problem [12], our approach is *not complete* in the sense that it does not always obtain a strategy which yields the required optimal probability. However, we define a scheme to *refine* the abstraction and extend the observability.

We implemented a `Python` tool–chain taking a graph formulation of the motion planning as input and applying the proposed abstraction–refinement procedure. The tool–chain uses `PRISM-games` [9] as a model checker for PGs. For the motion planning scenario considered, our preliminary results indicate an improvement of orders in magnitude over the state of the art in POMDP verification [13].

Related work: Sampling-based methods for motion planning in POMDP scenarios are considered in [14]–[17]. An overview on point-based value iteration for POMDPs is given

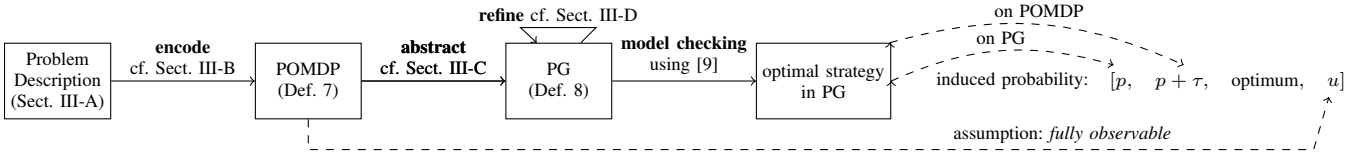

Fig. 1: Schematic overview of the approach

in [5]. Other methods employ control techniques to synthesize strategies with safety considerations under observation and dynamics noise [2], [18], [19]. Preprocessing of POMDPs in motion planning problems for robotics is suggested in [20].

General verification problems for POMDPs and their decidability have been studied in [21], [22]. A recent survey about decidability results and algorithms for ω-regular properties is given in [12], [23]. The probabilistic model checker PRISM has recently been extended to support POMDPs [13]. Partly based on the methods from [24], it produces lower and upper bounds for a variety of queries. Reachability can be analyzed for POMDPs for up to 30,000 states. In [25], an iterative refinement is proposed to solve POMDPs: Starting with total information, strategies that depend on unobservable information are excluded. In [26], a compositional framework for reasoning about POMDPs is introduced. Refinement based on counterexamples is considered in [27]. Partially observable probabilistic games have been considered in [28]. Finally, an overview of applications for PGs is given in [29].

II. Formal Foundations

A. Probabilistic games

For a finite or countably infinite set X, let $\mu\colon X \to [0,1]$ such that $\sum_{x\in X} \mu(x) = 1$ denote a *probability distribution* over X and $Dist(X)$ the set of all probability distributions over X. The *Dirac distribution* $\delta_x \in Dist(X)$ on $x \in X$ is given by $\delta_x(x) = 1$ and $\delta_x(y) = 0$ for $y \neq x$.

Definition 1 (Probabilistic game) *A* probabilistic game (PG) *is a tuple* $\mathsf{G} = (S_1, S_2, s_{init}, Act, P)$ *where* $S = S_1 \dot\cup S_2$ *is a finite set of* states*,* S_1 *the states of* Player 1*,* S_2 *the states of* Player 2*,* $s_{init} \in S$ *the* initial state*,* Act *a finite set of* actions*, and* $P\colon S \times Act \nrightarrow Dist(S)$ *a (partial)* probabilistic transition function*.*

Let $Act(s) = \{\alpha \in Act \mid (s,\alpha) \in \mathrm{dom}(P)\}$ denote the *available actions* in $s \in S$. We assume that PG G does not contain any deadlock states, i. e., $Act(s) \neq \emptyset$ for all $s \in S$. A *Markov decision process* (MDP) M is a PG with $S_2 = \emptyset$. We write $\mathsf{M} = (S, s_{\text{init}}, Act, P)$. A *discrete-time Markov chain* (MC) is an MDP with $|Act(s)| = 1$ for all $s \in S$.

The game is played as follows: In each step, the game is in a unique state $s \in S$. If it is a Player 1 state (i. e., $s \in S_1$), then Player 1 chooses an available action $\alpha \in Act(s)$ non-deterministically; otherwise Player 2 chooses. The successor state of s is determined probabilistically according to the probability distribution $P(s,\alpha)$: The probability of s' being the next state is $P(s,\alpha)(s')$. The game is then in state s'.

A path through G is a finite or infinite sequence $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots$, where $s_0 = s_{\text{init}}$, $s_i \in S$, $\alpha_i \in Act(s_i)$, and $P(s_i,\alpha_i)(s_{i+1}) > 0$ for all $i \in \mathbb{N}$. The $(i{+}1)$-th state s_i on π is $\pi(i)$, and $\mathrm{last}(\pi)$ denotes the last state of π if π is finite. The set of (in)finite paths is $Paths_{\mathsf{G}}^{\text{fin}}$ ($Paths_{\mathsf{G}}^{\text{inf}}$).

To define a probability measure over the paths of a PG G, the non-determinism needs to be resolved by *strategies*.

Definition 2 (PG strategy) *A* strategy σ *for* G *is a pair* $\sigma = (\sigma_1, \sigma_2)$ *of functions* $\sigma_i\colon \{\pi \in Paths_{\mathsf{G}}^{\text{fin}} \mid \mathrm{last}(\pi) \in S_i\} \to Dist(Act)$ *such that for all* $\pi \in Paths_{\mathsf{G}}^{\text{fin}}$*,* $\{\alpha \mid \sigma_i(\pi)(\alpha) > 0\} \subseteq Act\big(\mathrm{last}(\pi)\big)$*.* Σ_{G} *denotes the set of all strategies of* G *and* Σ_{G}^i *all Player-*i *strategies of* G*.*

For MDPs, the strategy consists of a Player-1 strategy only. A Player-i strategy σ_i is *memoryless* if $\mathrm{last}(\pi) = \mathrm{last}(\pi')$ implies $\sigma_i(\pi) = \sigma_i(\pi')$ for all $\pi, \pi' \in \mathrm{dom}(\sigma_i)$. It is *deterministic* if $\sigma_i(\pi)$ is a Dirac distribution for all $\pi \in \mathrm{dom}(\sigma_i)$. A *memoryless deterministic* strategy is of the form $\sigma_i\colon S_i \to Act$.

A strategy σ for a PG resolves all non-deterministic choices, yielding an *induced MC*, for which a *probability measure* over the set of infinite paths is defined by the standard cylinder set construction [30]. These notions are analogous for MDPs.

B. Partial observability

For many applications, not all system states are observable [2]. For instance, an agent may only have an estimate on the current state of its environment. In that case, the underlying model is a partially observable Markov decision process.

Definition 3 (POMDP) *A* partially observable Markov decision process (POMDP) *is a tuple* $\mathsf{D} = (\mathsf{M}, \mathcal{O}, \lambda)$ *such that* $\mathsf{M} = (S, s_{init}, Act, P)$ *is the* underlying MDP of D*,* $\mathcal{O}$ *a finite set of* observations*, and* $\lambda\colon S \to \mathcal{O}$ *the* observation function*.*

W. l. o. g. we require that states with the same observations have the same set of enabled actions, i. e., $\lambda(s) = \lambda(s')$ implies $Act(s) = Act(s')$ for all $s, s' \in S$. More general observation functions λ have been considered in the literature, taking into account the last action and providing a distribution over $\mathcal{O}$. There is a polynomial transformation of the general case to the POMDP definition used here [23].

The notions of paths and probability measures directly transfer from MDPs to POMDPs. We lift the observation function to paths: For a POMDP D and a path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots s_n \in Paths_{\mathsf{D}}^{\text{fin}}$, the associated *observation sequence* is $\lambda(\pi) = \lambda(s_0) \xrightarrow{\alpha_0} \lambda(s_1) \xrightarrow{\alpha_1} \cdots \lambda(s_n)$. Note that several paths in the underlying MDP M can give rise to the same observation sequence. Strategies have to take this restricted observability into account.

Definition 4 (Observation-based strategy) *An* observation-based strategy *of POMDP* D *is a function* $\sigma\colon Paths_{\mathsf{D}}^{\text{fin}} \to$

$Dist(Act)$ such that σ is a strategy for the underlying MDP and for all paths $\pi, \pi' \in Paths_{\mathsf{D}}^{\text{fin}}$ with $\lambda(\pi) = \lambda(\pi')$ we have $\sigma(\pi) = \sigma(\pi')$. $\Sigma_{\mathsf{D}}^{\text{o}}$ denotes the set of such strategies.

That means an observation-based strategy selects based on the observations and actions made along the current path.

The semantics of a POMDP can be described using a *belief MDP* with an uncountable state space. The idea is that each state of the belief MDP corresponds to a distribution over the states in the POMDP. This distribution is expected to correspond to the probability to be in a specific state based on the observations made so far. Initially, the belief is a Dirac distribution on the initial state. A formal treatment of belief MDPs is beyond the scope of this paper, for details see [5].

C. Specifications

Given a set $G \subseteq S$ of *goal states* and a set $B \subseteq S$ of *bad states*, we consider quantitative reach-avoid properties of the form $\varphi = \mathbb{P}_{\geqslant p}(\neg B \; \mathcal{U} \; G)$. The specification φ is satisfied by a PG if Player 1 has a strategy such that for all strategies of Player 2 the probability is at least p to reach a goal state without entering a bad state in between. For POMDPs, φ is satisfied if the agent has an observation-based strategy which leads to a probability of at least p.

For MDPs and PGs, memoryless deterministic strategies suffice to prove or disprove satisfaction of such specifications [31]. For POMDPs, observation-based strategies in their full generality are necessary [32].

III. Methodology

We first intuitively describe the problem and list the assumptions we make. After formalizing the setting, we present a formal problem statement. We present the intuition behind the concept of game-based abstraction for MDPs, how to apply it to POMDPs, and state the correctness of our method.

A. Problem Statement

We consider a motion planning problem that involves $n+1$ moving agents inside a *world* such as a landscape or a room. One agent is *controllable* (Agent 0), the other agents (also called *opponents*) move stochastically according to a fixed randomized strategy which is based on their own location and the location of Agent 0. We assume that all agents move in an alternating manner. A *position* of an agent defines the *physical location* inside the world as well as additional properties such as the agent's *orientation*. A graph models all possible movements of an agent between positions, referred to as the *world graph* of an agent. Therefore, nodes in the graph uniquely refer to *positions* while multiple nodes may refer to the same *physical location* in the world. We require that the graph does not contain any deadlocks: For every position, there is at least one edge in the graph corresponding to a possible action an agent can execute.

A *collision* occurs, if Agent 0 shares its location with another agent. The set of *goal nodes* (goal positions) in the graph is uniquely defined by a set of physical goal locations in the world. The *target* is to move Agent 0 towards a goal node without colliding with other agents. Technically, we need to synthesize a strategy for Agent 0 that maximizes the probability to achieve the target. Additionally, we assume:

- The strategies of all opponents are known to Agent 0.
- Agent 0 is able to observe its own position and knows the goal positions it has to reach.
- The positions of opponents are *observable* for Agent 0 from its current position, if they are *visible* with respect to a certain distance metric.

Generalizing the problem statement is discussed in Sect. VI.

B. Formal setting

We first define an individual world graph for each Agent i with $0 \leq i \leq n$ over a fixed set Loc of (physical) locations.

Definition 5 (World graph of Agent i) *The* world graph *G_i for Agent i over Loc is a tuple $G_i = (V_i, v_i^0, Mov_i, E_i, \ell_i)$ such that V_i is the set of* positions *and $v_i^0 \in V_i$ is the* initial position *of Agent i. Mov_i is the set of* movements[1]*; the edges $E_i \colon V_i \times Mov_i \nrightarrow V_i$ are the* movement effects*. The function $\ell_i \colon V_i \to Loc$ maps a position to the corresponding location.*

The *enabled movements* for Agent i in position v are $Mov_i(v) = \{m \in Mov_i \mid (v, m) \in \mathrm{dom}(E_i)\}$.

For Agent 0 we need the possibility to restrict its viewing range. This is done by a function $\nu_0 \colon V_0 \to 2^{Loc}$ which assigns to each position of Agent 0 the set of *visible locations*. According to our assumptions, for all $v \in V_0$ it holds that $\ell(v) \in \nu_0(v)$ and $Mov_i(v) \neq \emptyset$.

Each Agent i with $i > 0$ has a randomized strategy $\sigma_i \colon V_0 \times V_i \to Dist(Mov_i)$, which maps positions of Agent 0 and Agent i to a distribution over enabled movements of Agent i. The world graphs for all agents with randomized strategies for the opponents are subsumed by a single *world POMDP*. We first define the underlying *world MDP* modeling the possible behavior of all agents based on their associated world graphs.

Definition 6 (World MDP) *Given world graphs $G_0, \ldots, G_n$, the induced* world MDP *$\mathsf{M} = (S, s_{init}, Act, P)$ is defined by $S = V_0 \times V_1 \times \cdots \times V_n \times \{0, \ldots n\}$, $s_{init} = (v_0^0, v_1^0, \ldots, v_n^0, 0)$, and $Act = Mov_0 \dot\cup \{\bot\}$. P is defined by:*

- *For $\alpha \in Mov_0(v_0)$ and $\breve{v} \in V_1 \times \cdots \times V_n$, we have $P\big((v_0, \breve{v}, 0), \alpha\big) = \delta_{(E_0(v_0,\alpha),\breve{v},1)}$.*
- *$P\big((v_0, \hat{v}, v_i, \breve{v}), i), \bot\big)\big((v_0, \hat{v}, v_i', \breve{v}, i + 1 \bmod n + 1)\big) = q$, with $\hat{v} \in V_1 \times \cdots \times V_{i-1}$, $\breve{v} \in V_{i+1} \times \cdots \times V_n$, $1 \leq i \leq n$ and $q = \sum_{\{m \in Mov_i \mid E_i(v_i,m) = v_i'\}} \sigma_i(v_0, v_i)(m)$.*
- *0 in all other cases.*

The first item in the definition of P translates each movement in the world graph of Agent 0 into an action in the MDP that connects states with probability one, i. e., a Dirac distribution is attached to each action. Upon taking the action, the position of Agent 0 changes and Agent 1 has to move next.

[1] We use movements to avoid confusion with actions in PGs.

The second item defines movements of the opponents. In each state where Agent i is moving next, the action $\perp$ reflecting this move is added. The outcome of $\perp$ is determined by σ_i and the fact that Agent $i{+}1$ moves next.

Definition 7 (World POMDP) *Let* M *be a world MDP for world graphs* $G_0, \dots, G_n$. *The* world POMDP $\mathsf{D} = (\mathsf{M}, \mathcal{O}, \lambda)$ *with* $\mathcal{O} = V_0 \times \bigtimes_{1 \le i \le n} (V_i \dot\cup \{\mp\})$ *and* λ *defined by:*

$$\lambda((v_0, \dots, v_n))_i = \begin{cases} v_i, & \text{if } \ell(v_i) \in \nu_0(v_0), \\ \mp, & \text{otherwise.} \end{cases}$$

Thus, the position of Agent i is observed iff the location of Agent i is visible from the position of Agent 0, and otherwise a dummy value $\mp$, which is referred to as *far away*, is observed.

Given a set $\mathit{GoalLocations} \subseteq \mathit{Loc}$, the mappings $\ell_i \colon V_i \to \mathit{Loc}$ are used to define the states corresponding to collisions and goal locations. In particular, we have $\mathit{Collision} = \big\{((v_0, \dots, v_n), j) \in S \,\big|\, \exists 1 \le i \le n.\, \ell_0(v_0) = \ell_i(v_i)\big\}$ and $\mathit{Goals} = \big\{((v_0, \dots, v_n), j) \,\big|\, \ell_0(v_0) \in \mathit{GoalLocations}\big\}$.

Formal problem statement: Given a world POMDP D for a set of world graphs $G_0, \dots, G_n$, a set of collision states *Collision*, and a set of goal states *Goals*, an observation-based strategy $\sigma \in \Sigma^{\mathrm{o}}_{\mathsf{D}}$ for D is *p-safe* for $p \in [0, 1]$, if $\mathbb{P}^{\sigma}_{\ge p}(\neg \mathit{Collision}\ \mathcal{U}\ \mathit{Goals})$ holds. We want to compute a p-safe strategy for a given $p \in [0, 1]$.

C. Abstraction

We propose an abstraction method for world POMDPs that builds on *game-based abstraction* (GBAR), originally defined for MDPs [10], [11].

GBAR for MDPs: For an MDP $\mathsf{M} = (S, s_{\text{init}}, \mathit{Act}, P)$, we assume a partition $\Pi = \{B_1, \dots, B_k\}$ of S, i. e., a set of non-empty, pairwise disjoint, subsets (called blocks) $B_i \subseteq S$ with $\bigcup_{i=1}^{k} B_i = S$. GBAR takes the partition Π and turns each block into an abstract state B_i; these blocks form the Player 1 states. Then $\mathit{Act}(B_i) = \bigcup_{s \in B_i} \mathit{Act}(s)$. To determine the outcome of selecting $\alpha \in \mathit{Act}(B_i)$, we add intermediate *selector-states* $\langle B_i, \alpha \rangle$ as Player 2 states. In the selector state $\langle B_i, \alpha \rangle$, emanating actions reflect the choice of the actual state the system is in at B_i, i. e., $\mathit{Act}(\langle B_i, \alpha \rangle) = B_i$. For taking an action $s \in B_i$ in $\langle B_i, \alpha \rangle$, the distribution $P(s, \alpha)$ is lifted to a distribution over abstract states:

$$P\big(\langle B_i, \alpha \rangle, s\big)\big(B_j\big) = \sum_{s' \in B_j} P(s, \alpha)(s').$$

The semantics of this PG is as follows: For an *abstract state* B_i, Player 1 (controllable) selects an action to execute. In *selector-states*, the Player 2 (adversary) selects the worst-case state from B_i where the selection was executed.

Applying GBAR to POMDPs: The key idea in GBAR for POMDPs is to merge states with equal observations.

Definition 8 (Abstract PG) *The* abstract PG *of POMDP* $\mathsf{D} = \big((S, s_{\mathit{init}}, \mathit{Act}, P), \mathcal{O}, \lambda\big)$ *is* $\mathsf{G} = (S_1, S_2, s'_{\mathit{init}}, \mathit{Act}', P', R')$ *with* $S_1 = \big\{\{s \in S \,|\, \lambda(s) = \lambda(s')\} \,\big|\, s' \in S\big\}$, $S_2 = \big\{\langle B, \alpha \rangle \,\big|\, B \in S_1 \wedge \alpha \in \mathit{Act}(B)\big\}$, $s'_{\mathit{init}} = B$ *s. t.* $s_{\mathit{init}} \in B$, *and* $\mathit{Act}' = S \dot\cup \mathit{Act}$.

The transition probabilities P' *are defined as follows:*

- $P'(B, \alpha) = \delta_{\langle B, \alpha \rangle}$ *for* $B \in S_1$ *and* $\alpha \in \mathit{Act}(B)$,
- $P'\big(\langle B, \alpha \rangle, s\big)\big(B'\big) = \sum_{s' \in B'} P(s, \alpha)(s')$ *for* $\langle B, \alpha \rangle \in S_2$, $s \in B$, *and* $B' \in S_1$,
- *and* 0 *in all other cases.*

By construction, Player 1 has to select the same action for all states in an abstract state. As the abstract states coincide with the observations, this means that we obtain an observation-based strategy for the POMDP. For the classes of properties we consider, a memoryless deterministic strategy suffices for PGs to achieve the maximal probability of reaching a goal state without collision [31]. We thus obtain an optimal strategy $\sigma \colon S_1 \to \mathit{Act}'$ for Player 1 in the PG which maps every abstract state to an action. As abstract states are constructed such that they *coincide with all possible observations in the POMDP* (see Def. 8), this means that σ maps every observation to an action.

Abstract world PG: We now connect the abstraction to our setting. For ease of presentation, we assume in the rest of this section that there is only one opponent agent, i. e., we have Agent 0 and Agent 1. Therefore, if Agent 0 sees an agent and moves, no additional agent will appear. Moreover, Agent 0 either knows the exact state, or does not know where the opponent is.

First, we call the abstract PG of the world POMDP the *abstract world PG*. In particular, the abstract states B_k in the world PG are either of the form $B_k = (v_0, v_1, i)$ or of the form $B_k = (v_0, \mp, i)$. with $i \in \{0, 1\}$. In the former, the opponent is visible and the agent has full knowledge, in the latter only the own position is known. Recall that $\mp$ is a dummy value for the distance referred to as *far away*. Furthermore, all states in an abstract state correspond to the same position of Agent 0. For abstract states with full knowledge, there is no non-determinism of Player 2 involved as these states correspond to a single state in the world POMDP.

Correctness: We show that a safe strategy for Player 1 induces a safe strategy for Agent 0. Consider therefore a path $B_0 \xrightarrow{\alpha_0} \langle B_0, \alpha_0 \rangle \xrightarrow{s \in B_0} B_1 \xrightarrow{\alpha_1} \dots B_n$ in the PG. This path is projected to the blocks: $B_0 \xrightarrow{\alpha_0} B_1 \xrightarrow{\alpha_1} \dots B_n$. The location of Agent 0 encoded in the blocks is independent of the choices by Player 2. The sequence of actions $\alpha_0 \alpha_1 \dots$ thus yields a unique path of positions of Agent 0 in its world graph. Thus, if the path in the PG reaches a goal state, the path induces a path in the POMDP which also reaches a goal state. Moreover, the worst-case behavior over-approximates the probability for the opponent to be in any location, and any collision is observable. Thus if there is a collision in the POMDP, then there is a collision in the PG.

Formally, for a deterministic memoryless strategy σ' in the abstract world PG the corresponding strategy σ in the POMDP is defined as $\sigma(s) = \sigma'(B)$ for $s \in B$.

Theorem 1 *Given a p-safe strategy in an abstract world PG, the corresponding strategy in the world POMDP is p-safe.*

The assessment of the strategy is conservative: A p-safe strategy in the abstract world PG may induce a corresponding strategy in the POMDP which is $p+\tau$-safe for some $\tau > 0$. In particular, applying the corresponding strategy to the original POMDP yields a discrete-time Markov chain (MC). This MC can be efficiently analyzed by, e. g., probabilistic model checking to determine the value of $p+\tau$. Naturally, the optimal scheduler obtained for the PG does not need to be optimal in the POMDP.

All positions where Agent 1 is visible yield Dirac distributions in the belief MDP, i. e., the successor states in the MDP depend solely on the action choice. These beliefs are represented as single states in the abstract world PG. The abstraction lumps for each position of Agent 0 all (uncountably many) other belief states together.

D. Refinement of the PG

In the GBAR approach described above, we remove relevant information for an optimal strategy. In particular, the behavior of Agent 1 (the opponent) is strengthened (over-approximated):

- We abstract probabilistic movements of Agent 1 outside of the visible area into non-determinism.
- We allow jumps in Agent 1's movements, i.e., Agent 1 may change position in the PG. This is impossible in the POMDP; these movements are called *spurious*.

If, due to the lack of this information, no safe strategy can be found, the abstraction needs to be *refined*. In GBAR for MDPs [11], abstract states are split heuristically, yielding a finer over-approximation. In our construction, we cannot split abstract states arbitrarily: This would destroy the one-to-one correspondence between abstract states and observations. We would thus obtain a partially observable PG, or equivalently, for a strategy in the PG the corresponding strategy in the original POMDP is no longer observation-based.

However, we can restrict the *spurious movements* of Agent 1 by taking the history of observations made along a path into account. We present three types of *history-based refinements.*

a) One-step history refinement: If Agent 0 moves to state s from where Agent 1 is no longer visible, we have $\lambda(s) = \mp$. Upon the next move, Agent 1 could thus appear *anywhere*. However, until Agent 1 moves, the belief MDP is still in a Dirac distribution; the positions where Agent 1 can appear are thus restricted. Similarly, if Agent 1 disappears, upon a turn of Agent 0 in the same direction, Agent 1 will be visible again. The *(one-step history) refined world PG* extends the original PG by additional states (v_0, v_1, i) where $v_1 \notin \nu_0(v_0)$, i. e., v_1 is not visible for Agent 0. These "far away" states are only reached from states with full information. Intuitively, although Agent 1 is invisible, its position is remembered for one step.

b) Multi-step history refinement: Further refinement is possible by considering longer paths. If we first observe Agent 1 at location x, then loose visibility for one turn, and then observe Agent 1 again at position y, then we know that either x and y are at most two moves apart or that such a movement is *spurious*. To encode the observational history into the states of the abstraction, we store the *last known position* of Agent 1, as well as the *number m of moves* made since then. We then only allow Agent 1 to appear in positions which are at most m moves away from the last known position. We can cap m by the diameter of the graph.

c) Region-based multi-step history refinement: As the refinement above blows up the state space drastically, we utilize a technique called *magnifying lens abstraction* [33]. Instead of single locations, we define *regions of locations* together with the information if Agent 1 could be present. After each move, we extend the possible regions by all neighbor regions.

More formally, the *(multi-step history) refined world PG* has a refined far-away value $\mp$: Given a partition of the positions of Agent 1, e. g., extracted from the graph structure, into sets $\mathcal{X} = \{X_1, \ldots, X_l\}$ with $\bigcup_{X \in \mathcal{X}} = V_1$ and $X_i \cap X_j = \emptyset$ for all $1 \leq i < j \leq l$. We define $\mp' \colon \mathcal{X} \to \{0, 1\}$. Abstract states now are either of the form (v_0, v_1, i) as before, or $(v_0, \mp', i)$. For singleton regions, this coincides with the method proposed above. Notice that this approach also offers some flexibility: If for instance two regions are connected only by the visible area, Agent 0 can assure wether Agent 1 enters the other region.

Correctness: First, a deterministic memoryless strategy σ' on a refined abstract world PG needs to be translated to a strategy σ for the original POMDP while p-safety is conserved. Intuitively, as the proposed refinement steps encode history into the abstract world PG, the strategy σ is not memoryless anymore but has a finite memory at most m according to the maximum number of moves that are observed.

Theorem 2 *A p-safe strategy in a refined abstract world PG has a corresponding p-safe strategy in the world POMDP.*

The proposed refinements eliminate spurious movements of Agent 1 from the original abstract world PG. Intuitively, the number of states where Player 2 may select states with belief zero (in the underlying belief MDP) is reduced. We thus only prevent paths that have probability zero in the POMDP. Vice versa, the refinement does not restrict the movement of Agent 0 and any path leading to a goal state still leads to one in the refinement. However, the behavior of Agent 1 is restricted, therefore, the probability of a collision drops. Intuitively, for the refined PG strategies can be computed that are at least as good as for the original PG.

Theorem 3 *If an abstract world PG has a p-safe strategy, then its refined abstract world PG has a p'-safe strategy with $p' \geq p$.*

E. Refinement of the Graph

The proposed approach cannot solve every scenario — the problem is undecidable [12]. Therefore, if the method fails to find a p-safe scheduler, we do not know whether there exists such a scheduler. With increased visibility, however, the maximal level of safety does not decrease in both the POMDP and the PG. To determine good spots for increased visibility, we can use the analysis results: Locations in which a collision occurs are most likely good candidates.

IV. Case Study and Implementation

A. Description

For our experiments, we choose the following scenario: A (controllable) Robot *R* and a Vacuum Cleaner *VC* are moving around in a two-dimensional grid world with static opaque obstacles. Neither *R* nor *VC* may leave the grid or visit grid cells occupied by a static obstacle. The position of *R* contains the cell C_R (the location) and a wind direction. *R* can move one grid cell forward, or turn by $90°$ in either direction without changing its location. The position of *VC* is determined solely by its cell C_{VC}. In each step, *VC* can move one cell in any wind direction. We assume that *VC* moves to all available successor cells with equal probability.

The sensors on *R* only sense *VC* within a *viewing range* r around C_R. More precisely, *VC* is visible iff $\|C_R - C_{VC}\|_\infty \leq r$ and there is no grid cell with a static obstacle on the straight line from C_R's center to C_{VC}'s center. That means, *R* can observe the position of the *VC* if *VC* is in the viewing range and *VC* is not hidden behind an obstacle. A refinement of the world is realized by adding additional cameras, which make cells visible independent of the location of *R*.

B. Tool-Chain

To synthesize strategies for the scenario described above, we implemented a tool-chain in `Python`. The input consists of the grid with the locations of all obstacles, the location of cameras, and the viewing range. As output, two `PRISM` files are created: A PG formulation of the abstraction including one-step history refinement, to be analyzed using `PRISM-games` [9], and the original POMDP for `PRISM-pomdp` [13]. For multi-step history refinement, additional regions can be defined.

The encoding of the PG contains a precomputed lookup-table for the visibility relation. The PG is described by two parallel processes running interleaved: One for Player 1 and one for Player 2. As only *R* can make choices, they are listed in Player 1 actions, while *VC*'s moves are stored in Player 2 actions. More precisely, the process for *R* contains its location, and the process for *VC* either contains its location or a *far-away value*. Then, Player 1 makes its decision, afterwards the outcome of the move *and* the outcomes of the subsequent move of *VC* are compressed into one step of Player 2.

V. Experiments

A. Experimental Setup

All experiments were run on a machine with a 3.6 GHz Intel® Core™ i7-4790 CPU and 16 GB RAM, running Ubuntu Linux 16.04. We denote experiments taking over 5400 s CPU time as time-out and taking over 10 GB memory as mem-out (MO). We considered several variants of the scenario described in IV-A. The Robot always started in the upper-left corner and had the lower-right corner as target. The *VC* started in the lower-right corner. In all variants, the view range was 3. We evaluated the following five scenarios:

SC1 Rooms of varying size without obstacles.

SC2 Differently sized rooms with a cross-shaped obstacle in the center, which scales with increasing grid size.

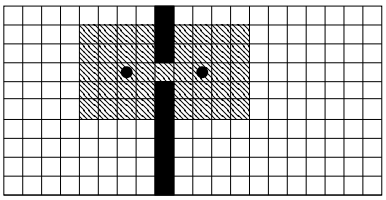

Fig. 2: Grid for **SC4**. The cameras observe the shaded area.

SC3 A 25×25 room with up to 70 randomly placed obstacles.

SC4 Two rooms (together 10×20) as depicted in Fig. 2. The doorway connecting the two rooms is a potential point of failure, as *R* cannot see to the other side. To improve reachability, we added cameras to improve visibility.

SC5 Corridors of the format $4 \times x$ – long, narrow grids that the Robot has to traverse from top to bottom, passing the *VC* on its way down.

B. Results

Table I shows the direct comparison between the POMDP description and the abstraction for **SC1**. The first column gives the grid size. Then, first for the POMDP and afterwards for the PG, the table lists the *number of states*, *non-deterministic choices*, and *transitions* of the model. The results include the safety probability induced by the optimal scheduler ("*Result*"), the run times (all in seconds) `PRISM` takes for *constructing the state space* from the symbolic description ("*Model Time*"), and finally the time to solve the POMDP / PG ("*Sol. Time*"). The last column shows the safety probability as computed using the fully observable MDP; it is an upper bound on the probability that is achievable for each grid. Note that optimal schedulers from this MDP are in general not observation-based and therefore not admissible for the POMDP. The time for creating the `PRISM` files was < 0.1 s in all cases.

Table II lists data for the PG constructed from **SC2** (first block of rows) and **SC5** (without additional refinement in the second block, with region-based multi-step history refinement in the third block), analogous to Table I. Additionally the runtime for creating the symbolic description is given ("Run times / Create"). On the fully observable MDP, the resulting probability is 1.0 for all **SC2**- and 0.999 for all **SC5** instances.

Table III shows the results for **SC3**. The first column ("#O") corresponds to the number of obstacles, while the remaining entries are analogous to Table II. The data for **SC4** is shown in Table IV. Its structure is identical to that of Table III, with the first column ("#C") corresponding to the number of cameras added for the graph refinement as in III-E.

C. Evaluation

Consider **SC1**: While for very small examples, `PRISM-pomdp` delivers results within reasonable time, already the 6×6 grid yields a mem-out. On the other hand, our abstraction handles grids up to 30×30 within minutes, while still providing schedulers with a solid performance. The safety probability is lower for small grids, as there is less room for *R* to avoid *VC*, and there are proportionally more situations in which *R* is trapped in a corner or against a wall. Notice that for the MDP, the state space for an $n \times n$ grid is in $\mathcal{O}(n^4)$ compared to a state space in $\mathcal{O}(r^2 n^2)$ for

TABLE I: Comparing the POMDP solution using `PRISM-pomdp` with the solution of the PG abstraction using `PRISM-games` on **SC1**.

	POMDP solution						PG solution						MDP
Grid size	States	Choices	Trans.	Result	Model Time	Sol. Time	States	Choices	Trans.	Result	Model Time	Sol. Time	Result
3×3	299	515	739	0.8323	0.063	0.26	400	645	1053	0.8323	0.142	0.036	0.8323
4×4	983	1778	2705	0.9556	0.099	1.81	1348	2198	3897	0.9556	0.353	0.080	0.9556
5×5	2835	5207	8148	0.9882	0.144	175.94	6124	10700	19248	0.9740	0.188	0.649	0.9882
5×6	4390	8126	12890	0.9945	0.228	4215.056	8058	14383	26079	0.9785	0.242	0.518	0.9945
6×6	6705	20086	12501	??	0.377	– MO –	10592	19286	35226	0.9830	0.322	1.872	0.9970
8×8	24893	47413	78338	??	1.735	– MO –	23128	81090	43790	0.9897	0.527	6.349	0.9998
10×10	66297	127829	214094	??	9.086	– MO –	40464	145482	78054	0.9914	0.904	6.882	0.9999
20×20	– Time out during model construction –						199144	745362	395774	0.9921	8.580	122.835	0.9999
30×30	– Time out during model construction –						477824	1808442	957494	0.9921	41.766	303.250	0.9999
40×40	– Time out during model construction –						876504	3334722	1763214	0.9921	125.737	1480.907	0.9999
50×50	– Time out during model construction –						1395184	5324202	2812934	0.9921	280.079	3129.577	– MO –

TABLE II: Results for the PG for differently sized models.

		PG				Run times		
	Grid	States	Choices	Trans.	Result	Create	Model	Solve
SC2	11×11	36084	66942	120480	0.9920	0.08	3	24
	21×21	173584	331482	618148	0.9972	1.19	41	103
	31×31	431044	834242	1572948	0.9977	7.62	231	312
	41×41	808504	1575402	2985348	0.9978	31.92	1220	805
SC5	4×40	50880	93734	170974	0.9228	0.01	1.4	17
	4×60	77560	143254	261534	0.8923	0.01	2.8	64
	4×80	104240	192774	352094	0.8628	0.01	5.2	110
	4×100	130920	242294	442654	0.8343	0.02	6.9	157
SC5 + ref.	4×40	55300	120848	198088	0.9799	0.01	25.2	38
	4×60	83820	182368	300648	0.9799	0.01	42.6	177
	4×80	112340	243888	403208	0.9799	0.01	74.2	191
	4×100	140860	305408	505768	0.9799	0.02	117.5	629

TABLE III: Results for **SC3**

	PG				Run times			MDP
#O	States	Choices	Trans.	Result	Create	Model	Solve	Result
10	297686	581135	1093201	0.9976	2.10	89.7	285.0	0.9999
40	234012	454652	823410	0.9706	2.74	87.3	179.1	0.9999
60	198927	385803	679321	0.6476	3.12	59.4	201.5	0.9999
70	187515	363401	633884	0.6210	3.30	59.4	116.1	0.9896

the PG, where r is the viewing range r. As a consequence, no upper bound could be computed for the 50×50 grid, as constructing the state space yielded a mem-out.

In Table II, for the **SC5** benchmarks, we see that the safety probability goes down for grids with a longer corridor. This is because in the abstraction, the Robot can meet the *VC* multiple times when traveling down the corridor. To avoid this unrealistic behavior, we used the region-based multi-step history refinement as described in Sect. III-D. Although we only look at histories of one step of the *VC* in length, this is enough to keep the safety probability at a value much closer to the upper bound, regardless of the length of the corridor.

Table II, **SC2**, indicates that the pre-computation of the visibility-lookup (see Sect. IV) for large grids with many obstacles eventually takes significant time, yet the model construction time increases on a faster pace. In comparison with **SC1**, we see that adding obstacles decreases the number of reachable states and thus also reduces the number of choices and transitions. Eventually, model construction takes longer than the actual model checking procedure.

Table III indicates that the model checking time is not significantly influenced by the number of obstacles. Furthermore, we observe that the first 50 obstacles behave benevolent and only marginally influence the safety probability, while at over 60 obstacles, the probability dips significantly compared to the upper bound. This is because the added obstacles provide blind spots, in which the Robot can no longer observe the movement of the *VC*.

TABLE IV: Results for **SC4**

	PG				Run times			MDP
#C	States	Choices	Trans.	Result	Create	Model	Solve	Result
none	76768	145562	271152	0.5127	0.22	7.9	23.5	0.9999
2	152920	291866	546719	0.9978	0.24	16.9	68.1	–

The same blind spot behavior can also be observed in Table IV (**SC4**). Here we add cameras to aid the robot by providing improved visibility around the blind spot, resulting in a near-perfect safety property. This doubles state space size and increases the model checking time by about 40 seconds.

VI. Discussion

Game-based abstraction successfully prunes the state space of MDPs by merging similar states. By adding an adversary that assumes the worst-case state, a PG is obtained. In general, this turns the POMDP at hand into a partially observable PG, which remains intractable. However, splitting according to observational equivalence leads to a fully observable PG. PGs can be analyzed by black-box algorithms as implemented, e. g., in `PRISM-games`, which also returns an optimal scheduler. The strategy from the PG can be applied to the POMDP, which yields the actual (higher) safety level.

In general, the abstraction can be too coarse; however, in the examples above, we have shown successfully that the game-based abstraction is not too coarse if one makes some assumptions about the POMDP. These assumptions are often naturally fulfilled by motion planning scenarios.

The assumptions from Sect. III-A can be relaxed in several respects: Our method naturally extends to *multiple opponents*. We restricted the method to a single controllable agent, but if information is shared among multiple agents, the method is applicable also to this setting. If information sharing is restricted, special care has to be taken to prevent information leakage. Richer classes of behavior for the opponents, including non-deterministic choices, are an important area for future research. This would lead to partially observable PGs, and game-based abstraction would yield three-player games.

As two sources of non-determinism are uncontrollable, both the opponents and the abstraction could be controlled by Player 2, thus yielding a PG again.

Supporting a richer class of temporal specifications is another option: `PRISM-games` supports a probabilistic variant of alternating (linear-) time logic extended by rewards and trade-off analysis. Using the same abstraction technique as we have presented, a larger class of properties thus can be analyzed. However, care has to be taken when combining invariants and reachability criteria arbitrarily, as they involve under- and over-approximations.

Our method can be generalized to POMDPs for other settings. We use the original problem statement on the graph only to motivate the correctness. The abstraction can be lifted (as indicated by Def. 8), for refinement, however, a more refined argument for correctness is necessary.

The proposed construction of the PG is straightforward and currently realized without constructing the POMDP first. This simplifies the implementation of the refinement, but mapping the scheduler on the POMDP is currently not supported. Improved tool support thus should yield better results (cf. the $(p + \tau)$-safety in Fig. 1) without changing the method.

VII. Conclusion

We utilized the successful technique of game-based abstraction to synthesize strategies for a class of POMDPs. Experiments show that this approach is promising. In future work, we will lift our approach to a broader class of POMDPs and improve the refinement steps, including an automatic refinement loop.

References

[1] R. A. Howard, *Dynamic Programming and Markov Processes*, 1st edition. The MIT Press, 1960.

[2] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, no. 1, pp. 99–134, 1998.

[3] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. The MIT Press, 2005.

[4] T. Wongpiromsarn and E. Frazzoli, "Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications," in *CDC*, IEEE, 2012, pp. 7644–7651.

[5] G. Shani, J. Pineau, and R. Kaplow, "A survey of point-based POMDP solvers," *Autonomous Agents and Multi-Agent Systems*, vol. 27, no. 1, pp. 1–51, 2013.

[6] J.-P. Katoen, "The probabilistic model checking landscape," in *LICS*, ACM, 2016, pp. 31–45.

[7] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: verification of probabilistic real-time systems," in *CAV*, ser. LNCS, vol. 6806, Springer, 2011, pp. 585–591.

[8] C. Dehnert, S. Junges, J.-P. Katoen, and M. Volk, "A storm is coming: a modern probabilistic model checker," *CoRR*, vol. abs/1702.04311, 2017.

[9] T. Chen, V. Forejt, M. Z. Kwiatkowska, D. Parker, and A. Simaitis, "PRISM-games: A model checker for stochastic multi-player games," in *TACAS*, ser. LNCS, vol. 7795, Springer, 2013, pp. 185–191.

[10] M. Kattenbelt and M. Huth, "Verification and refutation of probabilistic specifications via games," in *FSTTCS*, ser. LIPIcs, vol. 4, Schloss Dagstuhl, 2009, pp. 251–262.

[11] M. Kattenbelt, M. Kwiatkowska, G. Norman, and D. Parker, "A game-based abstraction-refinement framework for Markov decision processes," *FMSD*, vol. 36, no. 3, pp. 246–280, 2010.

[12] K. Chatterjee, M. Chmelík, and M. Tracol, "What is decidable about partially observable Markov decision processes with ω-regular objectives," *J. Comput. Syst. Sci.*, vol. 82, no. 5, pp. 878–911, 2016.

[13] G. Norman, D. Parker, and X. Zou, "Verification and control of partially observable probabilistic systems," *Real-Time Systems*, 2017, To appear.

[14] S. Patil, G. Kahn, M. Laskey, J. Schulman, K. Goldberg, and P. Abbeel, "Scaling up Gaussian belief space planning through covariance-free trajectory optimization and automatic differentiation," in *Algorithmic Foundations of Robotics XI*, ser. Springer Tracts in Advanced Robotics, vol. 107, Springer, 2014, pp. 515–533.

[15] B. Burns and O. Brock, "Sampling-based motion planning with sensing uncertainty," in *ICRA*, IEEE, 2007, pp. 3313–3318.

[16] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *ICRA*, IEEE, 2011, pp. 723–730.

[17] C.-I. Vasile, K. Leahy, E. Cristofalo, A. Jones, M. Schwager, and C. Belta, "Control in belief space with temporal logic specifications," in *CDC*, IEEE, 2016, pp. 7419–7424.

[18] K. Hauser, "Randomized belief-space replanning in partially-observable continuous spaces," in *Algorithmic Foundations of Robotics IX*, Springer, 2010, pp. 193–209.

[19] M. P. Vitus and C. J. Tomlin, "Closed-loop belief space planning for linear, Gaussian systems," in *ICRA*, IEEE, 2011, pp. 2152–2159.

[20] D. K. Grady, M. Moll, and L. E. Kavraki, "Extending the applicability of POMDP solutions to robotic tasks," *IEEE Trans. Robotics*, vol. 31, no. 4, pp. 948–961, 2015.

[21] L. de Alfaro, "The verification of probabilistic systems under memoryless partial-information policies is hard," DTIC Document, Tech. Rep., 1999.

[22] K. Chatterjee, M. Chmelík, R. Gupta, and A. Kanodia, "Qualitative analysis of POMDPs with temporal logic specifications for robotics applications," in *ICRA*, IEEE, 2015, pp. 325–330.

[23] ——, "Optimal cost almost-sure reachability in POMDPs," *Artif. Intell.*, vol. 234, pp. 26–48, 2016.

[24] H. Yu and D. P. Bertsekas, "Discretized approximations for POMDP with average cost," in *UAI*, AUAI Press, 2004, p. 519.

[25] S. Giro and M. N. Rabe, "Verification of partial-information probabilistic systems using counterexample-guided refinements," in *ATVA*, ser. LNCS, vol. 7561, Springer, 2012, pp. 333–348.

[26] X. Zhang, B. Wu, and H. Lin, "Assume-guarantee reasoning framework for MDP-POMDP," in *CDC*, IEEE, 2016, pp. 795–800.

[27] ——, "Counterexample-guided abstraction refinement for POMDPs," *CoRR*, vol. abs/1701.06209, 2017.

[28] K. Chatterjee and L. Doyen, "Partial-observation stochastic games: how to win when belief fails," *ACM Trans. Comput. Log.*, vol. 15, no. 2, 16:1–16:44, 2014.

[29] M. Svorenová and M. Kwiatkowska, "Quantitative verification and strategy synthesis for stochastic games," *Eur. J. Control*, vol. 30, pp. 15–30, 2016.

[30] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[31] A. Condon, "The complexity of stochastic games," *Inf. Comput.*, vol. 96, no. 2, pp. 203–224, 1992.

[32] S. M. Ross, *Introduction to Stochastic Dynamic Programming*. Academic Press, Inc., 1983, ISBN: 0125984200.

[33] L. de Alfaro and P. Roy, "Magnifying-lens abstraction for Markov decision processes," in *CAV*, ser. LNCS, vol. 4590, Springer, 2007, pp. 325–338.