Probabilistic Model Checking and Autonomy

Marta Kwiatkowska,[1] Gethin Norman[2] and David Parker[3]

[1]Department of Computer Science, University of Oxford, UK, OX1 3QD; email: marta.kwiatkowska@cs.ox.ac.uk
[2]School of Computing Science, University of Glasgow, UK, G12 8RZ; email: gethin.norman@glasgow.ac.uk
[3]School of Computer Science, University of Birmingham, UK; email: d.a.parker@cs.bham.ac.uk

Posted with permission from the Annual Review of Control, Robotics, and Autonomous Systems, Volume 5; copyright 2022 Annual Reviews, https://www.annualreviews.org/

Keywords

probabilistic modelling, temporal logic, model checking, strategy synthesis, stochastic games, equilibria

Abstract

Design and control of autonomous systems that operate in uncertain or adversarial environments can be facilitated by formal modelling and analysis. Probabilistic model checking is a technique to automatically verify, for a given temporal logic specification, that a system model satisfies the specification, as well as to synthesise an optimal strategy for its control. This method has recently been extended to multi-agent systems that exhibit competitive or cooperative behaviour modelled via stochastic games and synthesis of equilibria strategies. In this paper, we provide an overview of probabilistic model checking, focusing on models supported by the PRISM and PRISM-games model checkers. This includes fully observable and partially observable Markov decision processes, as well as turn-based and concurrent stochastic games, together with associated probabilistic temporal logics. We demonstrate the applicability of the framework through illustrative examples from autonomous systems. Finally, we highlight research challenges and suggest directions for future work in this area.

1. INTRODUCTION

As autonomous systems become embedded within computing infrastructure, from information systems through to security and robotics, there is a growing need for methodologies that ensure their safe, secure, reliable, timely and resource efficient execution. Design of computer systems can be facilitated by formal modelling and verification, and in particular *model checking*, which aims to automatically check if a system model satisfies given requirements typically expressed in temporal logic. *Autonomy*, however, creates additional demands of controllability, since autonomous systems operate in uncertain or adversarial environments, and strategic reasoning, to ensure effective coordination of cooperative or competitive behaviour of system components (agents).

Probabilistic model checking is a collection of techniques for the modelling of systems that exhibit probabilistic and non-deterministic behaviour, which supports not only their model checking against temporal logic, but also *synthesis of optimal controllers (strategies)* from temporal logic specifications. Probability is used to quantify environmental uncertainty and stochasticity, while non-determinism represents model decisions. Markov decision processes (MDPs) are typically employed to model and reason about the strategic behaviour of an agent against a stochastic environment, where specifications are expressed in probabilistic extensions of the temporal logics CTL or LTL. Partially observable Markov decision processes (POMDPs) permit similar modelling and analysis, but for contexts where the agent has limited power to observe its environment.

MDPs and POMDPs, however, are unable to faithfully represent the behaviour of multiple players competing or cooperating to achieve their individual goals. To this end, we employ multi-agent systems modelled via *stochastic games* and reason about their *strategic behaviour* for both zero-sum and nonzero-sum (equilibria) properties. For zero-sum properties, the utilities of an agent are the negation of the utility of its opponent, whereas for nonzero-sum each agent is pursuing its own quantitative objective. Probabilistic model checking has been recently extended to encompass both turn-based and concurrent stochastic games, together with an extension of the temporal logic that inherits the coalition operator from ATL, as well as synthesis of optimal *Nash equilibria* strategies (more precisely, subgame-perfect social welfare optimal strategies).

In this paper, we provide an overview of recent advances in probabilistic model checking, focusing on the model checking and strategic reasoning methods implemented in the PRISM (1) and PRISM-games (2) tools for discrete probabilistic models. The review covers fully observable and partially observable Markov decision processes (Sections 2 and 3 respectively), as well as turn-based and concurrent stochastic games (Sections 4 and 5 respectively), together with associated probabilistic temporal logics. We discuss the core types of quantitative analyses available for each model, as well as extensions such as multi-objective analysis and continuous-time, also called real-time in the model checking literature, models (Section 6). We demonstrate the applicability of the framework through illustrative examples, with emphasis on the areas of robotics and autonomy. Finally, we highlight challenges and suggest directions for future work in this area (Section 7).

2. MARKOV DECISION PROCESSES

We begin with Markov decision processes (MDPs) (3), which are a classic model for decision making under uncertainty. This is a discrete-time model, with discrete sets of states and actions, that allows both *non-determinism*, e.g., to represent the choices made by the

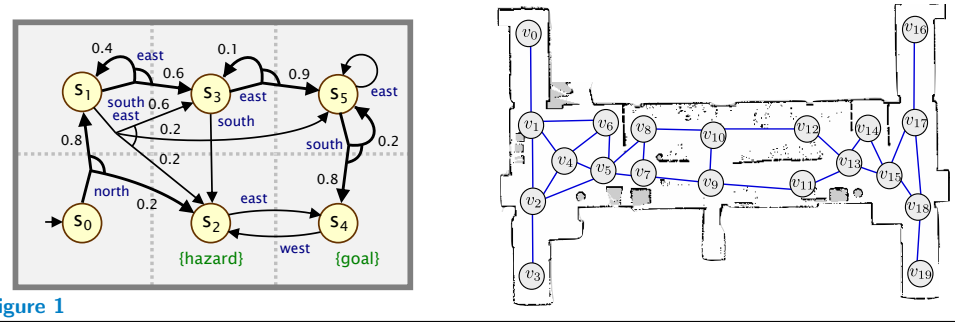

Figure 1

Left: A simple MDP representing a robot navigating through a grid; a (deterministic, memoryless) optimal policy for the property $\mathsf{P}_{\max=?}[\neg \mathsf{hazard}\ \mathsf{U}\ \mathsf{goal}]$ is marked in bold. Right: A topological map from (4) used to build a similar style MDP modelling a mobile robot exploring a building.

controller of a robot or vehicle, and discrete *probabilistic* choice, to model environmental uncertainty arising due to, for instance, the presence of humans, noisy sensors, unreliable communication media or faulty hardware.

We give a formal definition of MDPs below. Here, and in the remainder of the paper, $Dist(X)$ denotes the set of (discrete) *probability distributions* over a finite set X, i.e., functions $\mu : X \to [0, 1]$ such that $\sum_{x \in X} \mu(x) = 1$.

Definition 1 (Markov decision process)**.** A *Markov decision process* (MDP) is a tuple $\mathsf{M} = (S, \bar{s}, A, \delta, AP, L)$ where:

- S is a finite set of states and $\bar{s} \in S$ is an initial state;
- A is a finite set of *actions*;
- $\delta : (S \times A) \to Dist(S)$ is a (partial) probabilistic transition function, mapping state-action pairs to probability distributions over S;
- AP is a set of atomic propositions and $L : S \to 2^{AP}$ is a state labelling function.

The execution of an MDP M proceeds as follows. When in a state s, there is a non-deterministic choice over the actions that are *available* in the state, defined as the actions $a \in A$ such that $\delta(s, a)$ is defined and denoted $A(s)$. It is assumed that the set of available actions is non-empty for every state. After an action $a \in A(s)$ has been chosen in s, it is performed and the probability of transitioning to state $s' \in S$ equals $\delta(s, a)(s')$.

Example 1. A simple example of an MDP is shown in **Figure 1** (left); it models the movement of a robot through locations in a 3×2 grid. Each state (s_i) represents a location and actions taken in states result in probabilistic transitions to other locations. For example, in state s_1 there is a choice between moving *east* and *southeast*; if *east* is chosen, then with probability 0.6 the robot moves east and with probability 0.4 the robot remains in its current location. Also shown are atomic propositions (goal and hazard) needed for property specification. **Figure 1** (right) shows a topological map used to build a larger, similar-style MDP modelling a mobile robot traversing locations within an office building (4).

A path of M is defined by an alternating sequence of action choices and transitions. More formally, a path is a finite or infinite sequence $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} \cdots$ such that $s_0 = \bar{s}$, $a_i \in A(s_i)$ and $\delta(s_i, a_i)(s_{i+1}) > 0$ for all $i \geqslant 0$. $FPaths_{\mathsf{M}}$ and $IPaths_{\mathsf{M}}$ denote the sets of finite and infinite paths of M, respectively.

We next introduce the notion of a *strategy* (often also called a *policy*) of an MDP M, which resolves the non-determinism present in M. In particular, strategies decide which actions to take in states of the MDP, depending on its execution to date.

Definition 2 (MDP strategy)**.** A *strategy* of an MDP M is a function $\sigma : \mathit{FPaths}_{\mathsf{M}} \to \mathit{Dist}(A)$ such that, if $\sigma(\pi)(a) > 0$, then $a \in A(\mathit{last}(\pi))$ where $\mathit{last}(\pi)$ is the final state of π.

The set of all strategies of M is denoted Σ_{M}. We classify a strategy $\sigma \in \Sigma_{\mathsf{M}}$ in terms of its use of *randomisation* and *memory*.

- **Randomisation**: σ is *deterministic* (or *pure*) if $\sigma(\pi)$ picks a single action with probability 1 for all finite paths π, and *randomised* otherwise.
- **Memory**: σ is *memoryless* if $\sigma(\pi)$ depends only on $\mathit{last}(\pi)$ for all finite paths π, and *finite-memory* if there are finitely many *modes* such that, for any π, $\sigma(\pi)$ depends only on $\mathit{last}(\pi)$ and the current mode, which is updated each time an action is performed; otherwise, it is *infinite-memory*.

Under a particular strategy, the behaviour of MDP M is fully probabilistic and we can reason about the probability of different events. For a strategy σ of M, we denote by $\mathit{FPaths}^{\sigma}_{\mathsf{M}}$ and $\mathit{IPaths}^{\sigma}_{\mathsf{M}}$ the set of finite and infinite paths that correspond to the choices of σ. Following (5), we can definite a probability measure $\mathit{Prob}^{\sigma}_{\mathsf{M}}$ over $\mathit{IPaths}^{\sigma}_{\mathsf{M}}$ that corresponds to the behaviour of the MDP under σ. Using this probability measure we can then also define, for a random variable $X : \mathit{IPaths}_{\mathsf{M}} \to \mathbb{R}$, the expected value $\mathbb{E}^{\sigma}_{\mathsf{M}}(X)$ of X under σ.

Random variables can be used to introduce a variety of quantitative properties of MDPs. This is often achieved by augmenting an MDP with reward structures (these can in some cases represent costs, but for consistency we will use the term rewards). Example applications of rewards include: the energy consumption of a device, the number of tasks completed by a robot or the number of packets lost by a communication protocol.

Definition 3 (MDP reward structure)**.** A *reward structure* for an MDP M is a tuple $r = (r_S, r_A)$, where $r_S : S \to \mathbb{R}_{\geqslant 0}$ is a *state reward function* and $r_A : (S \times A) \to \mathbb{R}_{\geqslant 0}$ is an *action reward function*.

2.1. Property Specifications for MDPs

In order to formally specify the required behaviour of a system modelled as an MDP, we use quantitative extensions of *temporal logic*. Below, we show a fragment of the logic used as the property specification language for the PRISM model checker (1), which we refer to here as *the PRISM logic*. This is based on the logics PCTL (probabilistic computation tree logic) (6) and LTL (linear temporal logic) (7), and also incorporates operators to specify expected reward properties (8).

Definition 4 (Property syntax)**.** The syntax for a core fragment of the PRISM logic is:

$$
\begin{array}{rcl}
\Phi & \coloneqq & \mathtt{P}_{\bowtie p}[\,\psi\,] \mid \mathtt{R}^{r}_{\bowtie q}[\,\rho\,] \\
\psi & \coloneqq & \phi \mid \neg\psi \mid \psi \wedge \psi \mid \mathtt{X}\,\psi \mid \psi\ \mathtt{U}^{\leqslant k}\ \psi \mid \psi\ \mathtt{U}\ \psi \\
\rho & \coloneqq & \mathtt{I}^{=k} \mid \mathtt{C}^{\leqslant k} \mid \mathtt{F}\ \phi \\
\phi & \coloneqq & \mathtt{true} \mid \mathtt{a} \mid \neg\phi \mid \phi \wedge \phi
\end{array}
$$

where $\mathtt{a} \in AP$ is an atomic proposition, $\bowtie \in \{<, \leqslant, \geqslant, >\}$, $p \in [0,1]$, r is a reward structure, $q \in \mathbb{R}_{\geqslant 0}$ and $k \in \mathbb{N}$.

Above, we assume that a property Φ for an MDP comprises a single probabilistic ($\mathtt{P}$) or reward ($\mathtt{R}$) operator. The syntax also includes path (ψ) and reward (ρ) formulae, both evaluated over paths, and propositional logic (ϕ) formulae, evaluated over states. The intuitive meaning of the $\mathtt{P}$ and $\mathtt{R}$ operators, from the initial state of an MDP, is:

- $\mathtt{P}_{\bowtie p}[\,\psi\,]$ – the probability of a path satisfying path formula ψ satisfies the bound $\bowtie p$;
- $\mathtt{R}^{r}_{\bowtie q}[\,\rho\,]$ – the expected value of reward formula ρ, under reward structure r, satisfies the bound $\bowtie q$.

A propositional formula ϕ is *satisfied* (or *holds*) in a state s if it evaluates to true in that state, where an atomic proposition $\mathtt{a}$ is true if s is labelled with $\mathtt{a}$ (i.e., $\mathtt{a} \in L(s)$) and the logical connectives ($\neg$, $\wedge$) are interpreted in the usual way.

For path formulae ψ, the core temporal operators are:

- $\mathtt{X}\,\psi$ (*next*) – ψ is satisfied in the next state;
- $\psi_1\ \mathtt{U}^{\leqslant k}\ \psi_2$ (*bounded until*) – ψ_2 is satisfied within k steps, and ψ_1 is satisfied until that point;
- $\psi_1\ \mathtt{U}\ \psi_2$ (*until*) – ψ_2 is eventually satisfied, and ψ_1 is satisfied until then.

As is standard in model checking, we use the equivalences $\mathtt{F}\ \psi \equiv \mathtt{true}\ \mathtt{U}\ \psi$ (*eventually*) and $\mathtt{G}\ \psi \equiv \neg\mathtt{F}\ \neg\psi$ (*always*). If we restrict the sub-formulae of a path formula to be atomic propositions, then we get the following common property classes:

- $\mathtt{F}\ \mathtt{a}$ (*reachability*) – eventually a stated labelled with $\mathtt{a}$ is reached;
- $\mathtt{G}\ \mathtt{a}$ (*invariance*) – $\mathtt{a}$ labels all states;
- $\mathtt{F}^{\leqslant k}\ \mathtt{a}$ (*step-bounded reachability*) – $\mathtt{a}$ labels a state within the first k steps;
- $\mathtt{G}^{\leqslant k}\ \mathtt{a}$ (*step-bounded invariance*) – $\mathtt{a}$ labels states for at least the first k steps.

Without this restriction, path formulae allow temporal operators to be nested. In fact the syntax of path formulae given in Definition 4 is that of linear temporal logic (LTL) (7). LTL can express a range of useful property classes, including:

- $\mathtt{G}\ \mathtt{F}\ \psi$ (*recurrence*) – ψ is satisfied infinitely often;
- $\mathtt{F}\ \mathtt{G}\ \psi$ (*persistence*) – eventually ψ is always satisfied;
- $\mathtt{G}\ (\psi_1 \rightarrow \mathtt{X}\,\psi_2)$ – whenever ψ_1 is satisfied, ψ_2 is satisfied in the next state;
- $\mathtt{G}\ (\psi_1 \rightarrow \mathtt{F}\ \psi_2)$ – whenever ψ_1 is satisfied, ψ_2 is satisfied in the future.

Finally, considering reward formulae ρ, the three key operators are:

- $\mathtt{I}^{=k}$ (*instantaneous reward*) – state reward at time step k;
- $\mathtt{C}^{\leqslant k}$ (*bounded cumulative reward*) – reward accumulated over k steps;
- $\mathtt{F}\ \phi$ (*reachability reward*) – reward accumulated until a state satisfying ϕ is reached.

Although omitted from the syntax here for simplicity, it is also common to generalise the third case and consider the expected reward accumulated until some *co-safe* LTL formula is satisfied. Intuitively, these are path formulae ψ whose satisfaction occurs within finite time; examples include $(\mathtt{F}\ \mathtt{a}_1) \wedge (\mathtt{F}\ \mathtt{a}_2)$ and $\mathtt{F}\ (\mathtt{a}_1 \wedge \mathtt{F}\ \mathtt{a}_2)$, which require states labelled with $\mathtt{a}_1$ and $\mathtt{a}_2$ to be reached, either in any order (first case) or in a specified order (second case).

2.2. Probabilistic Model Checking of MDPs

Probabilistic model checking is an automated technique for constructing probabilistic models such as MDPs and then analysing them against behavioural specifications expressed in temporal logic. It can be used either to *verify* that a specification is always satisfied, regardless of any adversarial behaviour, or to *synthesise* a strategy under whose control the system's behaviour can be guaranteed to satisfy a specification.

These ideas are formalised below for the PRISM logic. We first require the following notation. Satisfaction of a path formula ψ can be represented by a random variable $X^{\psi} : \mathit{IPaths}_{\mathsf{M}} \to \mathbb{R}$ where $X^{\psi}(\pi) = 1$ if path π satisfies ψ and 0 otherwise. For a reward structure r and formula ρ, the random variable $X^{r,\rho} : \mathit{IPaths}_{\mathsf{M}} \to \mathbb{R}$ is such that $X^{r,\rho}(\pi)$ equals the state reward or accumulated reward corresponding to r and ρ for path π.

Verifying that an MDP M satisfies a formula Φ, denoted $\mathsf{M} \models \Phi$, is defined as follows.

Definition 5 (Verification problem for MDPs)**.** The *verification* problem is: given an MDP M and a formula Φ, verify whether $\mathsf{M} \models \Phi$, defined as:

$$\begin{array}{lcl} \mathsf{M} \models \mathtt{P}_{\bowtie p}[\,\psi\,] & \Leftrightarrow & \forall \sigma \in \Sigma_{\mathsf{M}}.\, \big(\, \mathbb{E}^{\sigma}_{\mathsf{M}}(X^{\psi}) \bowtie p \,\big) \\ \mathsf{M} \models \mathtt{R}^{r}_{\bowtie q}[\,\rho\,] & \Leftrightarrow & \forall \sigma \in \Sigma_{\mathsf{M}}.\, \big(\, \mathbb{E}^{\sigma}_{\mathsf{M}}(X^{r,\rho}) \bowtie q \,\big)\,. \end{array}$$

In practice, we often solve a *numerical* verification problem: given an MDP M, formula $\mathtt{P}_{\mathrm{opt}=?}[\,\psi\,]$ or $\mathtt{R}^{r}_{\mathrm{opt}=?}[\,\rho\,]$, where $\mathrm{opt} \in \{\min, \max\}$, compute $\mathbb{E}^{\mathrm{opt}}_{\mathsf{M}}(X)$ where $X = X^{\psi}$ or $X = X^{r,\rho}$, respectively, and:

$$\mathbb{E}^{\min}_{\mathsf{M}}(X) \stackrel{\mathrm{def}}{=} \inf_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(X) \quad \text{and} \quad \mathbb{E}^{\max}_{\mathsf{M}}(X) \stackrel{\mathrm{def}}{=} \sup_{\sigma \in \Sigma_{\mathsf{M}}} \mathbb{E}^{\sigma}_{\mathsf{M}}(X)\,.$$

Closely related is the strategy synthesis problem.

Definition 6 (Strategy synthesis problem for MDPs)**.** The *strategy synthesis* problem is: given an MDP M and formula Φ of the form $\mathtt{P}_{\bowtie p}[\,\psi\,]$ or $\mathtt{R}^{r}_{\bowtie q}[\,\rho\,]$, find a strategy $\sigma \in \Sigma_{\mathsf{M}}$ such that Φ is satisfied in M under σ, i.e., such that $\mathbb{E}^{\sigma}_{\mathsf{M}}(X^{\psi}) \bowtie p$ or $\mathbb{E}^{\sigma}_{\mathsf{M}}(X^{r,\rho}) \bowtie q$, respectively.

The *numerical* strategy synthesis problem is: given M and a formula of the form $\mathtt{P}_{\mathrm{opt}=?}[\,\psi\,]$ or $\mathtt{R}^{r}_{\mathrm{opt}=?}[\,\rho\,]$, where $\mathrm{opt} \in \{\min, \max\}$, find an *optimal* strategy $\sigma^{\star} \in \Sigma_{\mathsf{M}}$ such that $\mathbb{E}^{\sigma^{\star}}_{\mathsf{M}}(X) = \mathbb{E}^{\mathrm{opt}}_{\mathsf{M}}(X)$ for $X = X^{\psi}$ or $X = X^{r,\rho}$, respectively.

For general path formulae, optimal strategies are *finite-memory* and *deterministic*. On the other hand, for some common cases (e.g., the probability or expected accumulated reward to reach a target), *memoryless deterministic* strategies are sufficient.

Example 2. Returning to the MDP from Example 1, verification-style queries using the PRISM logic include:

- $\mathtt{P}_{\geqslant 0.8}[\,\mathtt{F}^{\leqslant 10}\mathsf{goal}\,]$ – under all possible strategies, the robot reaches its goal location within 10 steps with probability at least 0.8;
- $\mathtt{R}^{r_{hazard}}_{\leqslant 1.5}[\,\mathtt{C}^{\leqslant 20}\,]$ – for all possible strategies, the expected number of times that the robot enters the hazard location within the first 20 steps is at most 1.5;

and examples of numerical queries include:

- $\mathtt{P}_{\max=?}[\,\neg\mathsf{hazard}\ \mathtt{U}\ \mathsf{goal}\,]$ – what is the maximum probability that the goal can be reached while avoiding the hazard location?
- $\mathtt{R}^{r_{steps}}_{\min=?}[\,\mathtt{F}\ \mathsf{goal}\,]$ – what is the minimum expected number of steps to reach the goal?

Above, we use the following reward structures: r_{steps}, which assigns 1 to all state-action pairs; and r_{hazard}, which assigns 1 to all states labelled with atomic proposition hazard.

2.3. Model Checking Algorithms

Probabilistic model checking for MDPs requires a combination of graph-based algorithms, automata-based methods and numerical computation. The main components of the model checking procedure require computing optimal probabilities for path formulae and optimal expected values for reward formulae. For the simplest of these cases (e.g., the probability or expected accumulated reward to reach a target), various standard techniques for MDPs can be used (3), including: solving a linear programming problem; policy iteration (which builds a sequence of strategies until an optimal one is reached); and value iteration (which computes increasingly precise approximations to the optimal probability or expected value). Of these, value iteration is most commonly used in probabilistic model checking tools, for scalability and performance reasons, but variants that offer sound guarantees on the accuracy of results have also been introduced, e.g., (9, 10), as well as methods that employ simulation and heuristics, e.g., (10, 11). For finite-horizon (i.e., step-bounded) formulas. computation of the required values involves a finite number of steps of value iteration.

For reward formula, graph-based precomputation is often also needed. For example, given a reachability reward formula, a graph-based analysis must first be performed to find the states that reach the target with probability 1 under either all or at least one strategy (depending on whether we are interested in the minimum or maximum expected value).

For more complex path formulae, i.e., full LTL, one must first build a deterministic Rabin automaton (DRA) representation of the path formula and then construct a product MDP consisting of the MDP under study and the DRA. Next, through graph analysis, we identify states of the product MDP for which the probability of satisfaction is 0 or 1, and the *maximal end components* of the product. Informally, an end component is a set of states for which, under at least one strategy, it is possible to remain in forever once entered and a maximal end component has no other end component as a subset. After this step, numerical computation is performed on the product in the usual way.

The overall complexity for model checking MDPs against the PRISM logic is doubly exponential in the formula and polynomial in the size of the MDP. However, if we restrict the sub-formulae of path formulae to be atomic formulae, then DRAs are not required and the complexity reduces to linear in the formula and polynomial in the size of the MDP. Further details on the techniques needed to analyse MDPs can be found in, e.g., (8, 12, 13) and in standard texts on MDPs (14, 3).

2.4. Extensions, Tools and Applications

We conclude our discussion of MDPs by surveying extensions to the basic model checking problems, available software and some practical applications.

2.4.1. Extensions. One important extension of probabilistic model checking is to *multi-objective model checking*. This concerns verifying the satisfaction of, or synthesising a strategy that satisfies, multiple properties. The first work in this area concerned multi-objective model checking and strategy synthesis of MDPs against conjunctions of probabilistic LTL specifications (15). The approach has since been extended to general Boolean combinations of LTL properties (15, 16) and to include reward formulae (16, 17). The synthesised strategies for multi-objective queries have two forms of (finite-)memory: the first corresponds to the satisfaction of the individual objectives and the second, when objectives include general path formulae, the progress towards the satisfaction of such objectives.

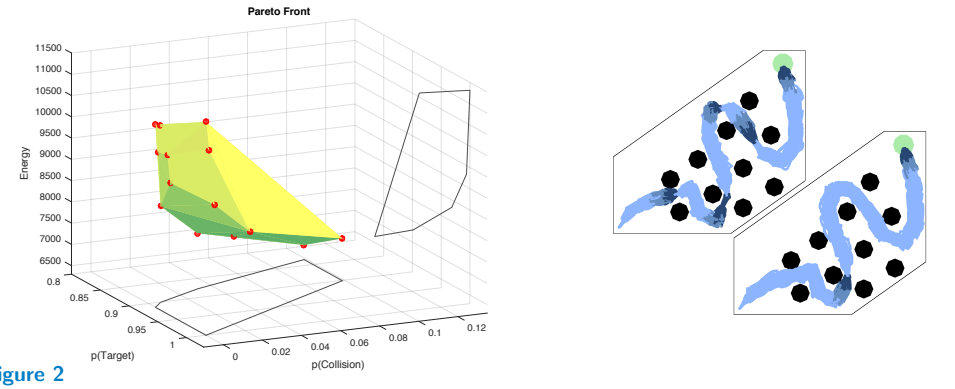

Figure 2

Left: Pareto curve, from (18), showing the trade-off between three objectives (collision avoidance, target reaching and energy consumption) for different localisation strategies along a particular trajectory of a mobile autonomous robot. Right: sampled executions for a synthesised strategy.

Multi-objective model checking has also been extended to numerical queries, which find the optimal value for one numerical objective when restricting to strategies that satisfy the remaining objectives (15, 16). In (17) this has been generalised to allow the analysis of the trade-offs between objectives by constructing the corresponding *Pareto curve*. **Figure 2** shows results from (18), which uses multi-objective probabilistic model checking of MDPs to study resource-performance trade-offs in mobile autonomous robots.

Another important extension incorporates parametric techniques. In this approach, one or more aspects of the MDP or specification under study, e.g. certain probabilities in the transition function or the step bound in a reward formula, are given as parameters. For Boolean-valued queries, *parameter synthesis* determines the set of parameter values for which the specification is satisfied. For numerical queries, *parametric model checking* returns a symbolic expression for the result, which is a function of the given parameters.

These techniques were first developed for models with only probabilistic behaviour, originally due to (19) and subsequently extended and optimised in (20) and (21), which represented transition probabilities as rational functions and applied language-theoretic techniques to return symbolic expressions for reachability probabilities. This approach has since been extended to MDPs (22) for a subclass of the PRISM logic. An alternative approach applied to MDPs is *parameter lifting* (23), where parametric transitions are replaced by non-deterministic choices over the extremal values. This non-determinism is placed under the control of a separate player, and therefore the analysis is then performed through probabilistic model checking of a two-player game (see Section 4).

Interval MDPs (24) generalise MDPs by having interval-valued transition probabilities, and therefore support modelling of systems when there is uncertainty or variation in the probabilistic behaviour. More general notions of such uncertain MDPs allow, for example, *convex uncertainty sets* to represent transition probabilities. Model checking algorithms have been developed for these models on a subset of the PRISM logic (25, 26), in a *robust* setting, i.e., where specifications are satisfied for any possible transition probabilities in the allowed set. Extensions to multi-objective queries also exist (27).

2.4.2. Tool support. A number of different software tools are available for model checking MDPs. Probably the most widely used is PRISM (1), which supports the logic of Defini-

tion 4 as well as both multi-objective specifications and parametric queries. The tool uses the PRISM modelling language, which is a simple, state-based language, based on Reactive Modules (28). STORM (29) is another tool that supports model checking of MDPs, for a subset of the logic in Definition 4, plus multi-objective and parametric extensions, and others such as long run average rewards and conditional probabilities. Models can be specified in a number of different modelling formalisms, including the PRISM language and JANI (30). Other general purpose probabilistic model checking tools include the Modest Toolset (31) and ePMC (32). PARAM (33) and PROPhESY (34) both offer tool support for parametric model checking and synthesis of MDPs.

2.4.3. Applications. Applications of MDP-based probabilistic model checking for autonomous systems include: motion planning (35, 36, 37), spacecraft reconfiguration (38), task allocation and planning for mobile robots (4, 39), analysis of the safety and reliability of robots in extreme environments (40), human-on-the-loop systems (41), robot battery charge scheduling (42) and autonomic computing (43). For a survey on using formal methods (including probabilistic model checking) for the verification of autonomous robotic systems see (44).

3. PARTIALLY OBSERVABLE MARKOV DECISION PROCESSES

Partially observable MDPs (POMDPs) extend MDPs by restricting the extent to which their current state can be observed, in particular by the strategies that control them. In the context of robotics, e.g., it may not be possible to accurately identify a robot's current location due to either limited precision or unreliability of their sensors. For security applications, participants in a protocol may rely on the use of private data.

Definition 7 (POMDP)**.** A POMDP is a tuple $\mathsf{P} = (S, \bar{s}, A, \delta, AP, L, \mathcal{O}, obs)$ where:

- $(S, \bar{s}, A, \delta, AP, L)$ is an MDP (see Definition 1);
- $\mathcal{O}$ is a finite set of *observations*;
- $obs : S \to \mathcal{O}$ is a labelling of states with observations;

such that $A(s) = A(s')$ for any states $s, s' \in S$ with $obs(s) = obs(s')$.

In a POMDP, the current state s cannot be directly determined; only the corresponding observation $obs(s) \in \mathcal{O}$ is known. Notice that Definition 7 requires observationally equivalent states to have the same available actions. This follows from the fact that states that have different sets of actions available would be observationally distinguishable as the available actions are not hidden, and hence should not have the same observations.

Above, we adopt a simple notion of observability, used in e.g. (45, 46), which is state-based and deterministic. More general notions of observations are also commonly used, and may depend on actions performed or are probabilistic. However, as demonstrated by (47), given a POMDP with these more general notions of observations, we can construct an equivalent (polynomially larger) POMDP of the form used here.

The notions of paths, strategies, probability measures and reward structures given in Section 2 for MDPs transfer directly to POMDPs. The one difference is that the set Σ_{P} of all strategies for a POMDP P only includes *observation-based strategies*.

Definition 8 (POMDP strategy)**.** A *strategy* of a POMDP $\mathsf{P} = (S, \bar{s}, A, \delta, AP, L, \mathcal{O}, obs)$ is a function $\sigma : FPaths_{\mathsf{P}} \to Dist(A)$ such that:

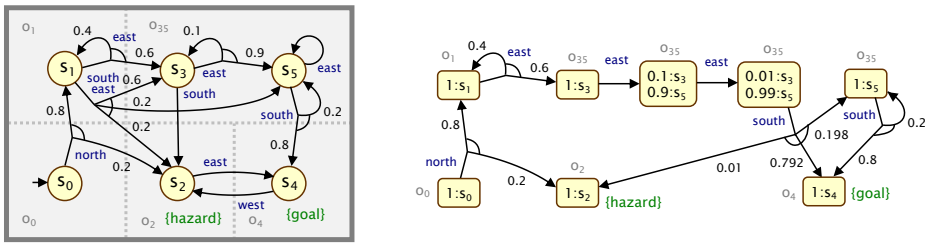

Figure 3

Left: POMDP variant of the MDP from **Figure 1**, where states s_3 and s_5 are observationally equivalent, and therefore cannot be distinguished by strategies. Right: Illustration of the POMDP under the control of a finite-memory strategy; states are labelled with the strategy's current belief as to its current state (as a probability distribution over states).

- σ is a strategy of the MDP $(S, \bar{s}, A, \delta, AP, L)$;
- for any paths $\pi = s_0 \xrightarrow{a_0} s_1 \xrightarrow{a_1} \cdots \xrightarrow{a_{n-1}} s_n$ and $\pi' = s'_0 \xrightarrow{a'_0} s'_1 \xrightarrow{a'_1} \cdots \xrightarrow{a'_{n-1}} s'_n$ satisfying $obs(s_i) = obs(s'_i)$ and $a_i = a'_i$ for all i, we have $\sigma(\pi) = \sigma(\pi')$.

Example 3. Figure 3 (left) shows a POMDP version of the MDP from Example 1 (**Figure 1**). The underlying states, transition probabilities and labels are identical, but we assume that the grid locations for states s_3 and s_5 are observationally indistinguishable due to localisation issues: these states map to the same observation (o_{35}), while other states have unique observations (o_i for s_i).

3.1. Model Checking for POMDPs

Properties for POMDPs can be specified using the same logic as for MDPs, presented in Section 2.1. The only change to the verification and strategy synthesis problems (see Definitions 5 and 6) is that the quantification is over observation-based strategies. However, probabilistic model checking for POMDPs is more challenging than for MDPs since the verification problem for core properties of the PRISM logic is undecidable (48).

Verification and strategy synthesis for POMDPs against finite-horizon problems, as well as discounted reward problems, is well studied in the fields of artificial intelligence and planning, and tool support exists, e.g., (49). However, the PRISM logic incorporates infinite-horizon properties such as unbounded probabilistic reachability ($\mathtt{P}_{\bowtie p}[\mathtt{F\ a}]$) and expected reward accumulated to reach a target ($\mathtt{R}^{r}_{\bowtie q}[\mathtt{F\ a}]$), without discounting, where the problem becomes undecidable (48). For further undecidability and complexity results of various POMDP model checking problems, see e.g., (45, 46).

Probabilistic model checking of POMDPs was proposed in (50) for a subset of the PRISM logic where path formulae only have propositional formulae as sub-formulae (i.e., without full LTL). The approach uses grid-based techniques (51, 52), which transform the POMDP under study to a fully observable *belief MDP* with uncountably many states and then approximate its solution based on a finite subset of states (grid points). Since the problem is undecidable, the approach only returns lower and upper bounds on the quantitative property of interest, and if the bounds are not precise enough, the grid can be refined and analysis repeated. The efficiency of this approach is improved in (53) using an abstraction-refinement loop to build smaller MDP approximations.

Strategy synthesis can be incorporated into these methods based on an analysis of the

belief MDP approximation. The resulting strategies are deterministic but require memory. Note that these methods assume a fixed initial state (or belief), in constrast to the methods for MDPs discussed above which can be performed for all states at once.

Similar methods have been extended to LTL queries (54) by translating such formulae to DRAs and using a product MDP construction. In the qualitative case (checking if the optimal probability equals 0 or 1), under the restriction to finite-memory strategies, model checking algorithms are given in (55) for LTL specifications.

Other approaches to POMDP model checking also work by imposing a limit on the memory available to strategies; this includes (56), which converts the problem to one of to parametric model checking, and (57) which uses a reduction to model checking for stochastic games using PRISM-games (2). A related method from (58) synthesises finite-memory POMDP strategies represented as recurrent neural networks.

Alternative methods include (47) which, under the requirement that all rewards in the POMDP are positive, extends approaches developed for finite-horizon objectives to approximate minimum expected reachability rewards. There is also (59), which uses counter-example-driven refinement to approximately solve MDPs in which components have partial observability of each other; and (60), which synthesises concurrent program constructs using a search over memoryless strategies in a partially observable stochastic game.

Example 4. Consider again the POMDP of Example 3 (**Figure 3**, left) and the property specification $\mathsf{P}_{\max=?}[\,\neg\mathsf{hazard}\ \mathsf{U}\ \mathsf{goal}\,]$. Any memoryless strategy (i.e., always choosing *south* or *east* in both s_3 and s_5) has zero probability of achieving this. **Figure 3** (right) illustrates a finite-memory strategy for the POMDP of Example 3 (**Figure 3**, left), which chooses *east* twice, increasing the chance of being in s_5, and then *south*. States are annotated with the current *belief*, i.e., the probability of being in each state.

3.2. Extensions, Tools and Applications

PRISM (1) implements the algorithms of (50) for a subset of the PRISM logic. STORM (29) also supports POMDP analysis, via the methods in (57, 56, 53). Extensions to synthesise robust strategies for *uncertain* POMDPs, as discussed earlier for MDPs, can be found in, e.g., (61). Applications of POMDP-based model checking for autonomous systems include robot motion planning (55, 57) and human-in-the-loop planning (62).

4. TURN-BASED STOCHASTIC GAMES

We now move beyond MDPs to *stochastic games*, which allow for the modelling of cooperative or competitive behaviour between multiple agents, in the presence of adversarial or uncertain environments. We start with *turn-based stochastic multi-player games* (TSGs), which have the same structure as MDPs, except that the states are partitioned amongst a set of players. Each state is controlled by one player, who resolves the action choices in that state. Formally, we have the following definition.

Definition 9 (Turn-based stochastic game)**.** A *turn-based stochastic (multi-player) game* (TSG) is a tuple $\mathsf{T} = (N, S, (S_i)_{i=1}^n, \bar{s}, A, \delta, AP, L)$, where:

- $(S, \bar{s}, A, \delta, AP, L)$ represents an MDP (see Definition 1);
- $N = \{1, \ldots, n\}$ is a finite set of *players*;
- $(S_i)_{i=1}^n$ is a partition of S.

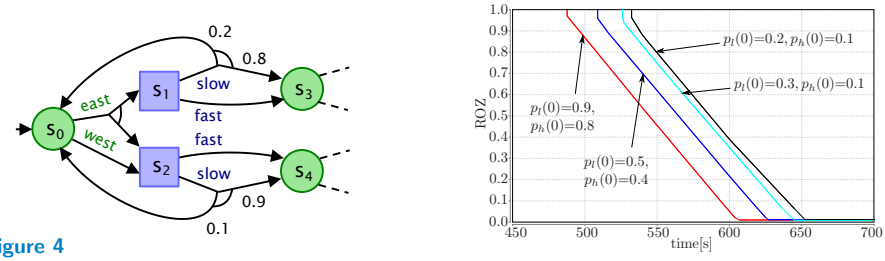

Figure 4

Left: A simple TSG modelling alternating decisions between a human operator and an autonomous robot. Right: Results from a more complex, but similar style TSG analysed in (63) for an unmanned aerial vehicle partially controlled by a human operator.

As for MDPs, in each state s of a TSG T, there is a set of available actions denoted $A(s)$, which are the actions a for which $\delta(s,a)$ is defined. However, in this case the choice of which available action is taken in s is under the control of a single player: the unique player $i \leqslant n$ such that $s \in S_i$. If player i selects action $a \in A(s)$ in s, then, as for MDPs, the probability of transitioning to state s' equals $\delta(s,a)(s')$.

The notion of paths and reward measures are the same as for MDPs. In the case of TSGs we do not have a single strategy, but instead a strategy for each player i of the TSG that resolves the choice of action in each state under the control of player i, based on the game's execution so far. Furthermore, to reason about the behaviour of a TSG we need a strategy for every player, called a *strategy profile*.

Definition 10 (TSG strategy)**.** A *strategy* of a TSG T is a function $\sigma_i : \{\pi \in \mathit{FPaths}_\mathsf{T} \mid \mathit{last}(\pi) \in S_i\} \to \mathit{Dist}(A)$ such that, if $\sigma_i(\pi)(a)>0$, then $a \in A(\mathit{last}(\pi))$. The set of all strategies of player $i \leqslant n$ is represented by Σ^i_T and a *strategy profile* is a tuple $\sigma = (\sigma_i)^n_{i=1}$ where $\sigma_i \in \Sigma^i_\mathsf{T}$ for all $i \leqslant n$.

Similarly to MDPs, for a TSG T and profile σ, we denote by $\mathit{FPaths}^\sigma_\mathsf{T}$ and $\mathit{IPaths}^\sigma_\mathsf{T}$ the set of finite and infinite paths of T that correspond to the choices made by the profile σ. Furthermore, for a given profile σ, we can define a probability measure $\mathit{Prob}^\sigma_\mathsf{T}$ over the set of infinite paths $\mathit{IPaths}^\sigma_\mathsf{T}$ and, for a random variable $X : \mathit{IPaths}_\mathsf{T} \to \mathbb{R}$, we can define the expected value $\mathbb{E}^\sigma_\mathsf{T}(X)$ of X under σ.

Example 5. Figure 4 (left) shows a fragment of a simple TSG modelling a human-robot system. Navigation decisions (*east* or *west*) are taken by a human operator (circular states, coloured green); then the robot decides autonomously how to follow these instructions (square states, coloured blue), here by choosing the speed (*slow* or *fast*) with which to proceed. **Figure** 4 (right) shows results from probabilistic model checking of a more complex TSG model in which an unmanned aerial vehicle performs surveillance under partial control of a human operator (63). It shows the trade-off between mission time and the likelihood of straying into "restricted operating zones" (ROZs) as operator accuracy varies.

4.1. Property Specifications for TSGs

To specify properties of TSGs, we consider an extension of the logic presented earlier for MDPs and POMDPs. This uses the *coalition* operator $\langle\langle C \rangle\rangle$ from alternating temporal logic (ATL) (64) to define *zero-sum* formulae. An extended version of this logic was presented as rPATL (and RPATL*) in (65).

Definition 11 (Property syntax for zero-sum games)**.** The syntax of extended PRISM logic for zero-sum games is:

$$\Phi \;\coloneqq\; \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\,\psi\,] \;\mid\; \langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\bowtie q}[\,\rho\,]$$

where path formulae ψ and reward formulae ρ are defined in identical fashion to the PRISM logic in Definition 4, $C \subseteq N$ is a coalition of players, $\bowtie \in \{<, \leqslant, \geqslant, >\}$, $p \in [0,1]$, r is a reward structure and $q \in \mathbb{R}_{\geqslant 0}$.

The zero-sum formulae $\langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\,\psi\,]$ and $\langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\bowtie q}[\,\rho\,]$ can be understood as specifying that the players in the coalition C can collectively ensure that the formula $\mathtt{P}_{\bowtie p}[\,\psi\,]$ or $\mathtt{R}^{r}_{\bowtie q}[\,\rho\,]$, respectively, is satisfied, against all possible strategies of the players in the set $N \setminus C$. In order to formalise the semantics of the extended PRISM logic, for a TSG T and coalition C, we denote by T_C the *coalition game*, that is, the 2-player TSG T_C in which the first player makes all the choices of all players in C and the second all players in $N \setminus C$.

When model checking TSGs, the verification and strategy synthesis problems coincide, since checking a property Φ reduces to showing that there exists a strategy for one coalition of players that satisfies a property for all strategies of another coalition.

Definition 12 (Verification and strategy synthesis problems for TSGs)**.** The *verification problem is*: given a TSG T and formula Φ, verify whether $\mathsf{T} \models \Phi$, defined as:

$$\begin{aligned}
\mathsf{T} \models \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\,\psi\,] \quad &\Leftrightarrow \quad \exists \sigma_1^\star \in \Sigma^1_{\mathsf{T}_C}.\, \big(\, \forall \sigma_2 \in \Sigma^2_{\mathsf{T}_C}.\, \mathbb{E}^{\sigma_1^\star,\sigma_2}_{\mathsf{T}_C}(X^\psi) \bowtie p \,\big) \\
\mathsf{T} \models \langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\bowtie q}[\,\rho\,] \quad &\Leftrightarrow \quad \exists \sigma_1^\star \in \Sigma^1_{\mathsf{T}_C}.\, \big(\, \forall \sigma_2 \in \Sigma^2_{\mathsf{T}_C}.\, \mathbb{E}^{\sigma_1^\star,\sigma_2}_{\mathsf{T}_C}(X^{r,\rho}) \bowtie q \,\big)
\end{aligned}$$

where T_C is the coalition game of T induced by C. The *strategy synthesis problem* is to find and return such a strategy $\sigma_1^\star$.

As for MDPs, in practice the *numerical verification problem* is often solved: given a TSG T and formula $\langle\!\langle C \rangle\!\rangle \mathtt{P}_{\mathrm{opt}=?}[\,\psi\,]$ or $\langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\mathrm{opt}=?}[\,\rho\,]$, where $\mathrm{opt} \in \{\min, \max\}$, compute:

$$val_{\mathsf{T}_{C^{\mathrm{opt}}}}(s, X) = \sup_{\sigma_1 \in \Sigma^1_{\mathsf{T}_C}} \inf_{\sigma_2 \in \Sigma^2_{\mathsf{T}_C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{T}_C}(X) \qquad 1.$$

where $C^{\mathrm{opt}} = C$ if $\mathrm{opt} = \max$ and equals $N \setminus C$ otherwise, and $X = X^\psi$ or $X = X^{r,\rho}$ respectively. The *numerical strategy synthesis* problem is to return a strategy $\sigma_1^\star \in \Sigma_{\mathsf{T}_{C^{\mathrm{opt}}}}$ such that $\inf_{\sigma_2 \in \Sigma^2_{\mathsf{T}_C}} \mathbb{E}^{\sigma^\star}_{\mathsf{T}_{C^{\mathrm{opt}}}}(X) = val_{\mathsf{T}_{C^{\mathrm{opt}}}}(X)$.

For general path formulae, optimal strategies are *finite-memory* and *deterministic*, while for infinite-horizon reward formulae and path formulae with only propositional formulae as sub-formulae *memoryless deterministic* optimal strategies exist.

As Definition 12 shows, for verifying TSGs, the main step is computing the value in Equation 1. If we consider the coalition game T_C as a *zero-sum game* (66), where the utility function of player 1 is the random variable X and the utility of the second[1] is $-X$, then it follows that this game is determined (67), and therefore the following equation holds:

$$\sup_{\sigma_1 \in \Sigma^1_{\mathsf{T}_C}} \inf_{\sigma_2 \in \Sigma^2_{\mathsf{T}_C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{T}_C}(X) = \inf_{\sigma_2 \in \Sigma^2_{\mathsf{T}_C}} \sup_{\sigma_1 \in \Sigma^1_{\mathsf{T}_C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{T}_C}(X) \qquad 2.$$

and Equation 1 is the *value* of this game (66). Furthermore, using Equation 2 we have the following equivalences:

$$\langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\,\psi\,] \;\equiv\; \langle\!\langle N \setminus C \rangle\!\rangle \mathtt{P}_{\neg(\bowtie p)}[\,\psi\,] \quad \text{and} \quad \langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\bowtie q}[\,\rho\,] \;\equiv\; \langle\!\langle N \setminus C \rangle\!\rangle \mathtt{R}^{r}_{\neg(\bowtie q)}[\,\rho\,].$$

[1] In a zero-sum game the utility of the second player is the negation of the first player's utility.

Example 6. Returning to the TSG of Example 5, PRISM logic queries for this include:

- $\langle\langle human \rangle\rangle \mathtt{P}_{\geqslant 0.6}[\,\neg \mathsf{crash}\ \mathtt{U}\ \mathsf{target}\,]$ – a human controller can ensure the robot reaches its target without crashing with probability at least 0.6, no matter what the robot does;
- $\langle\langle rbt \rangle\rangle \mathtt{R}^{r_{battery}}_{\max=?}[\,\mathtt{I}^{=10}\,]$ – what is the maximum expected battery level that the robot can ensure after 10 steps, no matter what the choices of the human controller are?
- $\langle\langle rbt \rangle\rangle \mathtt{R}^{r_{steps}}_{\geqslant 3.2}[\,\mathtt{F}\ \mathsf{target}\,]$ – the expected time that the robot requires to reach the target is at least 3.2, no matter what choices the human controller makes.

4.2. Model Checking Algorithms for TSGs

Model checking TSGs against the extended PRISM logic can be performed in a similar manner to MDPs (see Section 2.3) using numerical methods, automata and graph-based analysis (65). For basic properties such as the probability or expected accumulated reward to reach a target, numerical computation can be performed using several methods including solving a quadratic programming problem, policy iteration and value iteration (68). As for MDPs, variants of value iteration that yield error guarantees have also been developed (69).

For the full logic, including LTL, we can translate path formulae to deterministic parity automata (DPAs) and solve a product model which is a stochastic two-player zero-sum parity game (70). Graph-based algorithms are presented in (71). Overall, model checking is doubly exponential in the formula and polynomial in the size of the TSG. Similarly to MDP model checking, when the sub-formulae of path formulae are restricted to propositional formulae (i.e., no LTL), then DPAs are not required and parity winning conditions are replaced with reachability objectives and the complexity reduces to NP $\cap$ coNP.

4.3. Extensions, Tools and Applications

Lastly, we discuss extensions, tools and practical applications for model checking of TSGs.

4.3.1. Extensions. Multi-objective model checking has also been developed for TSGs, e.g., (72, 73) gives algorithms for the synthesis of ε-optimal strategies for TSGs which almost surely satisfy conjunctions of mean payoffs, ratio rewards and Boolean combinations of expected mean-payoffs. On the other hand, (74) concerns synthesising strategies of TSGs that almost surely surely maintain the averages of a number of long-run average reward specifications remaining above a given multi-dimensional threshold vector.

4.3.2. Tools and applications. PRISM-games (2) enables the modelling and analysis of TSGs against the extended PRISM logic (see Definition 11) and multi-objective specifications. Applications using TSGs and PRISM-games to model autonomous systems include autonomous urban driving (72), smart grids (65), human-in-the-loop planning (75, 76), managing collections of autonomic systems (77, 78) and self-adaption (79). In addition, GIST (80) allows the analysis of ω-regular properties of TSGs and GAVS+ (81) is a general-purpose tool for algorithmic game solving including TSGs.

5. CONCURRENT STOCHASTIC GAMES

In this section, we generalise TSGs to *concurrent stochastic games* (CSGs), in which players choose their actions simultaneously in each state and without already knowing the actions

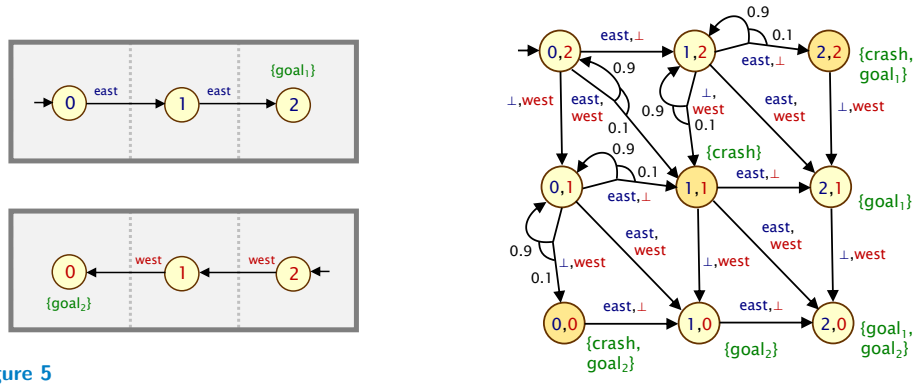

Figure 5

CSG model of two robots moving through a 3×1 grid. Left: the transitions of robot 1 (blue) and robot 2 (red). Right: CSG representing their concurrent execution.

being taken by other players. This can provide a more realistic model of interactive autonomous agents operating concurrently.

Definition 13 (Concurrent stochastic game)**.** A *concurrent stochastic multi-player game* (CSG) is a tuple $\mathsf{C} = (N, S, \bar{s}, A, \Delta, \delta, AP, L)$ where:

- $(S, \bar{s}, A, \delta, AP, L)$ represents an MDP (see Definition 1);
- $N = \{1, , \ldots, n\}$ is a finite set of players;
- $A = (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$ where A_i is a finite set of actions available to player $i \leqslant n$, $A_i \cap A_j = \varnothing$ for all $i \neq j$ and $\bot$ is an idle action disjoint from the set $\cup_{i=1}^{n} A_i$;
- $\Delta \colon S \to 2^{\cup_{i=1}^{n} A_i}$ is an action assignment function.

As previously, we specify available actions for a CSG, where for each state s and player i the set of available actions, denoted $A_i(s)$, equals $\Delta(s) \cap A_i$ if this set is non-empty, and $\{\bot\}$ otherwise. If each player $i \leqslant n$ selects action $a_i \in A_i(s)$ in state s, then the probability of transitioning to state s' equals $\delta(s, (a_1, , \ldots, a_n))(s')$.

The notions of paths and reward measures are the same as for MDPs. Similarly to TSGs, there is not a single strategy but instead a strategy for each player i of the CSG that resolves the choices of that player.

Definition 14 (CSG Strategy)**.** A *strategy* for player i in a CSG C is a function of the form $\sigma_i \colon \mathit{FPaths}_{\mathsf{C}} \to \mathit{Dist}(A_i \cup \{\bot\})$ such that, if $\sigma_i(\pi)(a_i) > 0$, then $a_i \in A_i(\mathit{last}(\pi))$. We denote by Σ_{C}^{i} the set of all strategies for player i and a *strategy profile* is a tuple $\sigma = (\sigma_i)_{i=1}^{n}$ where $\sigma_i \in \Sigma_{\mathsf{C}}^{i}$ for all $i \leqslant n$.

As for the previous models, given a CSG C and profile σ, we denote by $\mathit{FPaths}_{\mathsf{C}}^{\sigma}$ and $\mathit{IPaths}_{\mathsf{C}}^{\sigma}$ the set of finite and infinite paths of C that correspond to the choices made by σ. Furthermore, for a given profile σ, we can define a probability measure $\mathit{Prob}_{\mathsf{C}}^{\sigma}$ over the set of infinite paths $\mathit{IPaths}_{\mathsf{C}}^{\sigma}$ and, for a random variable $X : \mathit{IPaths}_{\mathsf{C}} \to \mathbb{R}$, we can define the expected value $\mathbb{E}_{\mathsf{C}}^{\sigma}(X)$ of X under σ.

Example 7. Figure 5 illustrates a CSG modelling two robots aiming to traverse the same 3×1 grid in opposite directions. On the left are the transitions of robot 1 (top, blue) and robot 2 (bottom, red). On the right is the CSG over the product state space (state (l_1, l_2)

is when robot i is in location l_i), where, if the robots attempt to move to the same grid point with probability 0.1, they both move and crash into each other[2], and with probability 0.9 they do not move. The states of the product are labelled with atomic propositions representing when the robots have reached their goals and if they have crashed.

5.1. Property Specifications for CSGs

We now extend the logic syntax previously defined for zero-sum games in Section 4.1. We add the specification of *nonzero-sum* properties, using the notion of equilibria, which allows players to have objectives that are distinct, but not necessarily directly opposing.

Definition 15 (Property syntax for zero-sum and nonzero-sum games)**.** The syntax of the extended PRISM logic for zero and nonzero-sum games is:

$$\begin{array}{rcl} \Phi & \coloneqq & \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\bowtie p}[\,\psi\,] \;\mid\; \langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\bowtie q}[\,\rho\,] \;\mid\; \langle\!\langle C_1 \mathrel{:} \cdots \mathrel{:} C_m \rangle\!\rangle_{\mathrm{opt}\bowtie q}(\theta) \\ \theta & \coloneqq & \mathtt{P}[\,\psi\,] + \cdots + \mathtt{P}[\,\psi\,] \;\mid\; \mathtt{R}^{r}[\,\rho\,] + \cdots + \mathtt{R}^{r}[\,\rho\,] \end{array}$$

where path formulae ψ and reward formulae ρ are defined in identical fashion to the PRISM logic in Definition 4, C and $C_1, \ldots, C_m$ are coalitions of players such that $C_i \cap C_j = \varnothing$ for all $1 \leqslant i \neq j \leqslant m$ and $\cup_{i=1}^{m} C_i = N$, $\mathrm{opt} \in \{\min, \max\}$, $\bowtie \in \{<, \leqslant, \geqslant, >\}$, $p \in [0,1]$, r is a reward structure and $q \in \mathbb{R}_{\geqslant 0}$.

The logic has been further extended with *nonzero-sum formulae* and sums of probabilistic and reward objectives. Nonzero-sum formulae take the form $\langle\!\langle C_1 \mathrel{:} \cdots \mathrel{:} C_m \rangle\!\rangle_{\mathrm{opt}\bowtie q}(\theta)$, where $C_1, \ldots, C_m$ are sets of coalitions that represent a partition of players N, and θ is either the sum $\mathtt{P}[\,\psi_1\,] + \cdots + \mathtt{P}[\,\psi_m\,]$ of m probabilistic objectives or the sum $\mathtt{R}^{r_1}[\,\rho_1\,] + \cdots + \mathtt{R}^{r_m}[\,\rho_m\,]$ of m reward objectives. We can consider the ith element in these sums as representing the objective X_i^θ for the coalition C_i, where $X_i^\theta = X^{\psi_i}$ or $X_i^\theta = X^{r_i,\rho_i}$, respectively (see Section 2.1). Their meaning is as follows. The formula $\langle\!\langle C_1 \mathrel{:} \cdots \mathrel{:} C_m \rangle\!\rangle_{\max\bowtie q}(\theta)$ is satisfied if there exists a profile $\sigma^\star$ such that:

- no coalition C_i for $i \in M$ can deviate from $\sigma^\star$ in order to *increase* their objective X_i^θ;
- there is no other such profile for which the sum of the objectives $(X_i^\theta)_{i=1}^m$ is *greater* than the sum under $\sigma^\star$;
- the sum of the objectives $(X_i^\theta)_{i=1}^m$ under $\sigma^\star$ satisfies $\bowtie q$;

and we call such a profile that satisfies the first two conditions a *social welfare optimal Nash equilibrium* (SWNE). The first condition is the standard definition of a Nash equilibrium (NE) profile (82) and the second additionally requires that the *sum* of the objectives is maximal, making it an SWNE. On the other hand, the formula $\langle\!\langle C_1 \mathrel{:} \cdots \mathrel{:} C_m \rangle\!\rangle_{\min\bowtie q}(\theta)$ is satisfied if there exists a profile $\sigma^\star$ such that:

- no coalition C_i for $i \in M$ can deviate from $\sigma^\star$ in order to *decrease* their objective X_i^θ;
- there is no other such profile for which the sum of the objectives $(X_i^\theta)_{i=1}^m$ is *less* than the sum under $\sigma^\star$;
- the sum of the objectives $(X_i^\theta)_{i=1}^m$ under $\sigma^\star$ satisfies $\bowtie q$;

and we call such a profile that satisfies the first two conditions a *social cost optimal NE* (SCNE). The first condition corresponds to the standard definition of a NE profile for the

[2]For simplicity, in this model we assume the robots do not crash when swapping grid points.

objectives $(-X_i^\theta)_{i=1}^m$ and the second is what is required for the profile to be social cost optimal. For further details and formal definitions see (83). To give formal semantics, for a CSG C and partition $\mathcal{C}$ of the players into m coalitions, we denote by $\mathsf{C}_\mathcal{C}$ the m-player *coalition game*, that is, the game C in which the ith player makes the choices for all players in coalition C_i.

We can now define the verification and strategy synthesis problems for CSGs which, as for TSGs, coincide as the verification problem reduces to demonstrating the existence of a certain profile. For these problems, we restrict our attention to subgame-perfect NE (84), which are NE in every state of the corresponding CSG.[3]

Definition 16 (Verification and strategy synthesis problems for CSGs)**.** The *verification problem is*: given a CSG C and formula Φ, verify whether $\mathsf{C} \models \Phi$, where for zero-sum formulae the satisfaction relation is the same as for TSGs (see Definition 12) and for nonzero-sum formulae we have:

$$\mathsf{C} \models \langle\!\langle C_1 : \cdots : C_m \rangle\!\rangle_{\mathrm{opt} \bowtie q}(\theta) \quad \Leftrightarrow \quad \exists \sigma^\star \in \Sigma_{\mathsf{C}_\mathcal{C}} . \left(\textstyle\sum_{i=1}^m \mathbb{E}^{\sigma^\star}_{\mathsf{C}_\mathcal{C}}(X_i^\theta) \right) \bowtie q$$

and $\sigma^\star$ is a subgame-perfect SWNE if opt = max, and a subgame-perfect SCNE if opt = min, for the objectives $(X_i^\theta)_{i=1}^m$ in the coalition game $\mathsf{C}_\mathcal{C}$. The *strategy synthesis problem* is then to return such a profile $\sigma^\star$.

The *numerical* verification and strategy synthesis problems for zero-sum formulae are as for TSGs (see Definition 12): for nonzero-sum formulae, given a CSG C and formula $\langle\!\langle C_1, \ldots, C_m \rangle\!\rangle_{\mathrm{opt}=?}[\theta]$ where $\mathrm{opt} \in \{\min, \max\}$, compute, for the objectives $(X_i^\theta)_{i=1}^m$, the sum $\sum_{i=1}^m \mathbb{E}^{\sigma^\star}_{\mathsf{C}_\mathcal{C}}(X_i^\theta)$ for a sub-game perfect SWNE profile $\sigma^\star$ if opt = max and for a SCNE profile $\sigma^\star$ otherwise, and then return $\sigma^\star$.

Optimal strategies are *finite-memory randomised*, and in both cases *memoryless randomised* strategies are sufficient when restricting to infinite-horizon properties with only propositional formulae as sub-formulae.

Example 8. Returning to the CSG of Example 7, specifications for this model include:

- $\langle\!\langle rbt_1 \rangle\!\rangle \mathtt{P}_{\max=?}[\, \neg \mathsf{crash}\ \mathtt{U}\ \mathsf{goal}_1 \,]$ – what is the maximum probability with which the first robot can ensure that it reaches its goal without crashing, regardless of the behaviour of the second robot;
- $\langle\!\langle rbt_2 \rangle\!\rangle \mathtt{R}^{r_{steps}}_{\leqslant 4.5}[\, \mathtt{F}\ \mathsf{goal}_2 \,]$ – there is a strategy for the second robot that can ensure its goal is reached within 4.5 expected steps, no matter the behaviour of the first robot;
- $\langle\!\langle rbt_1 : rbt_2 \rangle\!\rangle_{\max \geqslant 2}(\mathtt{P}[\, \mathtt{F}\ \mathsf{goal}_1 \,] + \mathtt{P}[\, \neg \mathsf{crash}\ \mathtt{U}^{\leqslant 10} \mathsf{goal}_2 \,])$ – the robots can collaborate so that both reach their goal with probability 1, with the additional condition that the second has to reach its goal within 10 steps and not crash;
- $\langle\!\langle rbt_1 : rbt_2 \rangle\!\rangle_{\min=?}(\mathtt{R}^{r_{steps}}[\, \mathtt{F}\ \mathsf{goal}_1 \,] + \mathtt{R}^{r_{steps}}[\, \mathtt{F}\ \mathsf{goal}_2 \,])$ – what is the sum of expected reachability values when the robots collaborate and each minimises the expected number of steps to reach their goal?

[3] Since the existence of NE is an open problem (85) for infinite-horizon properties, while ε-NE profiles have been shown to exist for any $\varepsilon > 0$, the definitions are in fact given in the context of a particular ε, see (83) for details.

5.2. Model Checking Algorithms for CSGs

For CSG model checking, only limited progress has been made to date. In the qualitative case, (86, 87) present graph-based algorithms for reachability properties and omega-regular languages (which can encode all LTL properties). In the quantitative case, (83) introduces model checking algorithms for the extended PRISM logic restricted to a subset of the logic in which the sub-formulae of path formula are propositional formulae and there are only two coalitions in nonzero-sum formulae. There are also restrictions on the class of CSGs that can be analysed, which can be viewed as a variant of *stopping games* (88).

The model checking algorithms presented in (83) involve graph-based analysis followed by value iteration. In the case of zero-sum games, during value iteration for each state, at each iteration, an LP problem of size $|A|$ must be solved (corresponding to finding the value of a zero-sum one-shot game), which has complexity PTIME (89). On the other hand, for nonzero-sum formulae, during value iteration for each state, at each iteration, all solutions to an LCP problem of size $|A|$ must be found (corresponding to finding all the NE of a nonzero-sum one-shot two-player game). It has been shown that the complexity of such problems is PPAD (*polynomial parity argument in a directed graph*) (90). Regarding the number of iterations required in either case, for finite-horizon objectives this is equal to the step bound in the formula. On the other hand, for infinite-horizon objectives, the number of iterations depends on the convergence criterion and an exponential lower bound has been shown in the worst-case (91).

This approach has since been extended (92) to allow any number of coalitions to appear in nonzero-sum formulae. In this case, during value iteration, for each state, at each iteration, one must find all the NE of a nonzero-sum one-shot m-player game, and it has been shown that finding all the NE when there are three (or more) players is PPAD-complete (93).

Other work related to CSGs and nonzero-sum properties includes: (94, 95), which study the existence and complexity of finding NE; (96), which analyses the complexity of finding subgame-perfect NE for reachability properties; and (97), which investigates the complexity of equilibrium design. The existence of stochastic equilibria with imprecise deviations and a PSPACE algorithm to compute such equilibria is considered in (98).

5.3. Tools and Applications

PRISM-games (2) supports the model checking of CSGs against a restricted class of the extended PRISM logic, where the only sub-formulae of path formulae are propositional formulae and nonzero-sum formula are restricted to two coalitions. An extension of PRISM-games that supports more general nonzero-sum formulae is presented in (92). Applications of CSG model checking to date include robotics, computer security and communication protocols such as Aloha (83).

6. FURTHER EXTENSIONS

In this section, we discuss some further extensions to the models, logics and model checking algorithms to broaden the range of systems and properties that can be analysed.

6.1. Continuous-Time Models

The models we have so far presented are all *discrete-time* models exhibiting both probabilistic and non-deterministic behaviour. However, for certain systems it is necessary to also

model continuous-time characteristics and the interplay between the continuous, discrete and stochastic dynamics.

Probabilistic timed automata (PTAs) (99, 100, 101) extend MDPs with continuous-time *clocks*, which are variables whose values range over the non-negative reals and increase at the same rate as time. POMDPs, TSG and CSGs have also been extended with continuous-time to POPTAs (102), TPTGs (103) and CPTGs (104), respectively. Reward structures for these models are again specified in terms of both state and action rewards, though state rewards now specify the *rate* at which rewards are accumulated as time passes in a state. The PRISM logic can also be applied to these continuous-time models, where the key difference is that the bounds appearing in path and reward formulae now correspond elapsed time, rather than the number of discrete steps. Additional continuous-time properties can be modelled by adding *formula clocks* and *freeze quantifiers* to the logic (100).

Model checking and strategy synthesis algorithms for these models are based on first constructing a finite-state discrete-time model (e.g., an MDP) and then performing model checking on this model. There are a number of approaches that can be employed, including:

- the region graph construction for PTAs (100);
- the boundary region graph for PTAs (105) and CPTGs (104);
- the digital clocks method for PTAs (102), POPTAs (102) and TPTGs (103);
- forwards reachability for PTAs (100);
- backwards reachability for PTAS (106, 107);
- abstraction refinement with stochastic games for PTAs (108).

Although this survey focuses on discrete-state models, we also mention briefly that probabilistic model checking techniques have been developed for models with *continuous* state, and for *hybrid* stochastic systems with both discrete and continuous aspects to their state space; see, for example, (109, 110).

6.2. Verification of Complex Systems

One of the most successful approaches to improve the scalability of non-probabilistic model checking of complex systems is through *abstraction-refinement* frameworks (111). This is based on first constructing a small model that abstracts aspects of the complex system that do not relate to the specification, while preserving the satisfaction of a given specification. This abstraction can then be verified and, if the abstraction indeed satisfies the specification, then so does the complex system. If the abstraction does not satisfy the specification, then information from the model checking process, which in the non-probabilistic case is usually a counter-example path, can then be used to either show that the specification is not satisfied by the complex system or, if this is not possible, to refine the abstraction until the satisfaction of specification by the complex system can be determined.

In the setting of probabilistic models the focus has been on MDPs. The first framework was introduced by (112, 113) using probabilistic simulations (114). Extending this, (115, 116) have developed a framework based on predicate abstraction (117) and probabilistic counter-examples (118) implemented in the PASS tool (119). An alternative refinement of MDPs is presented in (120) using TSGs for the abstract model to maintain a distinction between the non-determinism of the original MDP and that introduced through the abstraction process. By maintaining this distinction, model checking of the abstract models yields separate lower and upper bounds for the given specification, and therefore a quantitative

measure of the quality of the abstraction.

An alternative approach to improve scalability is *compositional* verification (121, 122), which allows the correctness of a system to be verified through the model checking of individual components in isolation.

7. CONCLUSIONS

We have provided an overview of probabilistic model checking techniques with an emphasis on autonomous system modelling, verification and strategy synthesis from temporal logic specifications. The described techniques have been implemented in the PRISM and PRISM-games probabilistic model checkers and used to model and analyse a variety of case studies from robotics, security and computer networks.

SUMMARY POINTS

1. Probabilistic model checking provides a unified framework and a formal language to facilitate the construction of a wide range of single- and multi-agent autonomous system models that operate in uncertain or adversarial environments, and whose agents cooperate or compete, and proceed concurrently or in turn-based fashion.
2. Temporal logic can be extended with suitable operators (probabilistic, coalition, reward) to conveniently specify a variety of quantitative, zero-sum or distinct, objectives of autonomous agents and supports high-level motion planning, strategic reasoning and coordination of their behaviours.
3. Optimal controllers (or strategies) for autonomous agents (or coalitions of agents), including equilibria, can be automatically synthesised from temporal logic specifications. For partially observable models and infinite-horizon objectives, it cannot be guaranteed that the controllers are optimal due to undecidability of the underlying problem. For multi-agent models, optimality is defined in terms of social welfare and may need to be weakened to ε-optimality for infinite-horizon objectives.

FUTURE ISSUES

1. Model checking of partially observable stochastic games has been studied (123). However, little progress has so far been made on developing practical, approximate verification and strategy synthesis algorithms.
2. Efficiency is still a major limitation when verifying complex real-world systems. Therefore it is essential to continue to improve scalability, which includes extending compositional probabilistic model checking to all models.
3. In the case of CSGs, the efficiency of equilibria computation of normal form games (82) is the main limitation. Enhancing the performance of this method is thus necessary, as is extending the computation to different notions of equilibria and games including: Stackelberg (124), correlated (125) and psychological games (126).
4. One limitation is that none of the presented models allows the modelling of agents that can learn and adapt, e.g. through reinforcement learning (127) or neuro-symbolic reasoning (128). This is an important issue that needs to be addressed.

ACKNOWLEDGMENTS

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 834115) and the EPSRC Programme Grant on Mobile Autonomy (EP/M019918/1).

DISCLOSURE STATEMENT

The authors are not aware of any affiliations, memberships, funding, or financial holdings that might be perceived as affecting the objectivity of this review.

LITERATURE CITED

1. Kwiatkowska M, Norman G, Parker D. 2011. *PRISM 4.0: verification of probabilistic real-time systems*. In *Proc CAV'11*, vol. 6806 of *LNCS*, pp. 585–591. Springer. prismmodelchecker.org
2. Kwiatkowska M, Norman G, Parker D, Santos G. 2020. *PRISM-games 3.0: stochastic game verification with concurrency, equilibria and time*. In *Proc. CAV'20*, vol. 12225 of *LNCS*, pp. 475–487. Springer. prismmodelchecker.org/games/
3. Puterman M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley and Sons
4. Lacerda B, Parker D, Hawes N. 2014. *Optimal and dynamic planning for Markov decision processes with co-safe LTL specifications*. In *Proc. IROS'14*, pp. 1511–1516. IEEE
5. Kemeny J, Snell J, Knapp A. 1976. *Denumerable Markov Chains*. Springer
6. Hansson H, Jonsson B. 1994. A logic for reasoning about time and reliability. *FAC* 6(5):512–535
7. Pnueli A. 1981. The temporal semantics of concurrent programs. *TCS* 13:45–60
8. Forejt V, Kwiatkowska M, Norman G, Parker D. 2011. *Automated verification techniques for probabilistic systems*. In *SFM'11*, vol. 6659 of *LNCS*, pp. 53–113. Springer
9. Haddad S, Monmege B. 2018. Interval iteration algorithm for MDPs and IMDPs. *TCS* 735:111–131
10. Brázdil T, Chatterjee K, Chmelík M, Forejt V, Křetínský J, et al. 2014. *Verification of Markov decision processes using learning algorithms*. In *Proc. ATVA'14*, vol. 8837 of *LNCS*, pp. 98–114. Springer
11. Křetínský J, Meggendorfer T. 2020. Of cores: a partial-exploration framework for Markov decision processes. *Logical Methods in Computer Science* 16(4)
12. Baier C, Katoen JP. 2008. *Principles of Model Checking*. MIT Press
13. de Alfaro L. 1997. Formal verification of probabilistic systems. Ph.D. thesis, Stanford University
14. Bellman R. 1957. *Dynamic Programming*. Princeton University Press
15. Etessami K, Kwiatkowska M, Vardi M, Yannakakis M. 2008. Multi-objective model checking of Markov decision processes. *LMCS* 4(4):1–21
16. Forejt V, Kwiatkowska M, Norman G, Parker D, Qu H. 2011. *Quantitative multi-objective verification for probabilistic systems*. In *Proc. TACAS'11*, vol. 6605 of *LNCS*, pp. 112–127. Springer
17. Forejt V, Kwiatkowska M, Parker D. 2012. *Pareto curves for probabilistic model checking*. In *Proc. ATVA'12*, vol. 7561 of *LNCS*, pp. 317–332. Springer
18. Lahijanian M, Svorenova M, Morye AA, Yeomans B, Rao D, et al. 2018. Resource-performance trade-off analysis for mobile robots. *IEEE RA-L* 3(3):1840–1847
19. Daws C. 2004. *Symbolic and parametric model checking of discrete-time Markov chains*. In *Proc. ICTAC'04*, vol. 3407 of *LNCS*, pp. 280–294. Springer

20. Hahn E, Hermanns H, Zhang L. 2011. Probabilistic reachability for parametric Markov models. *STTT* 13(1):3–19
21. Jansen N, Corzilius F, Volk M, Wimmer R, Ábrahám E, et al. 2014. *Accelerating Parametric Probabilistic Verification*. In *Proc. QEST'14*, pp. 404–420
22. Hahn E, Han T, Zhang L. 2011. *Synthesis for PCTL in parametric Markov decision processes*. In *Proc. NFM'11*, vol. 6617 of *LNCS*. Springer
23. Quatmann T, Dehnert C, Jansen N, Junges S, Katoen JP. 2016. *Parameter synthesis for Markov models: faster than ever*. In *Proc. ATVA'16*, vol. 9938 of *LNCS*, pp. 50–67. Springer
24. Givan R, Leach S, Dean T. 2000. Bounded-parameter Markov decision processes. *Artif. Intell.* 122(1-2):71–109
25. Wolff E, Topcu U, Murray R. 2012. *Robust control of uncertain Markov decision processes with temporal logic specifications*. In *Proc. CDC'12*, pp. 3372–3379
26. Puggelli A, Li W, Sangiovanni-Vincentelli A, Seshia S. 2015. *Polynomial-time verification of PCTL properties of MDPs with convex uncertainties*. In *Proc. CAV'13*, vol. 8044 of *LNCS*, pp. 527–542. Springer
27. Hahn E, Hashemi V, Hermanns H, Lahijanian M, Turrini A. 2019. Interval Markov decision processes with multiple objectives: from robust strategies to Pareto curves. *ACM Trans. Model. Comput. Simul.* 29(27):1–31
28. Alur R, Henzinger T. 1999. Reactive modules. *FMSD* 15(1):7–48
29. Dehnert C, Junges S, Katoen JP, Volk M. 2017. *A Storm is coming: A modern probabilistic model checker*. In *Proc. CAV'17*, vol. 10427 of *LNCS*, pp. 592–600. stormchecker.org
30. Budde C, Dehnert C, Hahn E, Hartmanns A, Junges S, Turrini A. 2017. *JANI: quantitative model and tool interaction*. In *Proc. TACAS'17*, vol. 10206 of *LNCS*, pp. 151–168. Springer. jani-spec.org
31. Hartmanns A, Hermanns H. 2014. *The Modest toolset: an integrated environment for quantitative modelling and verification*. In *Proc. TACAS'14*, vol. 8413 of *LNCS*, pp. 593–598. Springer. modestchecker.net
32. Hahn EM, Li Y, Schewe S, Turrini A, Zhang L. 2014. *iscasMc: a web-based probabilistic model checker*. In *Proc. FM'14*, vol. 8442 of *LNCS*, pp. 312–317. Springer. iscasmc.ios.ac.cn/IscasMC
33. Hahn E, Hermanns H, Wachter B, Zhang L. 2010. *PARAM: a model checker for parametric Markov models*. In *Proc. CAV'10*, vol. 6174 of *LNCS*, pp. 660–664. Springer. depend.cs.uni-saarland.de/tools/param/
34. Dehnert C, Junges S, Jansen N, Corzilius F, Volk M, et al. 2015. *PROPhESY: A PRObabilistic ParamEter SYnthesis tool*. In *Proc. CAV'15*, vol. 9206 of *LNCS*, pp. 214–231. Springer. moves.rwth-aachen.de/research/tools/prophesy/
35. Lahijanian M, Andersson S, Belta C. 2012. Temporal logic motion planning and control with probabilistic satisfaction. *IEEE Trans. Robot.* 28(2):396–409
36. Lahijanian M, Andersson S, Belta C. 2015. Formal verification and synthesis for discrete-time stochastic systems. *IEEE Trans. Automat. Contr.* 60(8):2031–2045
37. Fraser D, Giaquinta R, Hoffmann R, Ireland M, Miller A, Norman G. 2020. Collaborative models for autonomous systems controller synthesis. *FAC* 32:157–186
38. Nardone V, Santone A, Tipaldi M, Glielmo L. 2016. *Probabilistic model checking applied to autonomous spacecraft reconfiguration*. In *Proc. MetroAeroSpace'16*, pp. 556–560. IEEE
39. Lacerda B, Faruq F, Parker D, Hawes N. 2019. Probabilistic planning with formal performance guarantees for mobile service robots. *Int. J. Robot. Res.* 38(19):1098–1123
40. Zhao X, Robu V, Flynn D, Dinmohammadi F, Fisher M, Webster M. 2019. *Probabilistic model checking of robots deployed in extreme environments*. In *Proc. AAAI'19*, vol. 33, pp. 8066–8074. AAAI Press
41. Li N, Adepu S, Kang E, Garlan D. 2020. *Explanations for human-on-the-loop: a probabilistic model checking approach*. In *Proc. SEAMS'20*, pp. 181–187. ACM
42. Tomy M, Lacerda B, Hawes N, Wyatt J. 2019. *Battery charge scheduling in long-life au-*

tonomous mobile robots. In *Proc. ECMR'19*, pp. 1–6. IEEE
43. Calinescu R, Ghezzi C, Kwiatkowska M, Mirandola R. 2012. Self-adaptive software needs quantitative verification at runtime. *Communications of the ACM* 55(9):69–77
44. Luckcuck M, Farrell M, Dennis L, Dixon C, Fisher M. 2019. Formal specification and verification of autonomous robotic systems: A survey. *ACM Comput. Surv.* (100):1–41
45. Baier C, Bertrand N, Größer M. 2008. *On decision problems for probabilistic Büchi automata*. In *Proc. FOSSACS'08*, vol. 4962 of *LNCS*, pp. 287–301. Springer
46. Chatterjee K, Chmelík M, Tracol M. 2013. *What is decidable about partially observable Markov decision processes with omega-regular objectives*. In *Proc. CSL'13*, pp. 165–180
47. Chatterjee K, Chmelík M, Gupta R, Kanodia A. 2016. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.* 234:26–48
48. Madani O, Hanks S, Condon A. 2003. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.* 147(1–2):5–34
49. Poupart P. 2005. Exploiting structure to efficiently solve large scale partially observable markov decision processes. Ph.D. thesis, University of Toronto
50. Norman G, Parker D, Zou X. 2015. *Verification and control of partially observable probabilistic real-time systems*. In *Proc. FORMATS'15*, vol. 9268 of *LNCS*, pp. 240–255. Springer
51. Lovejoy W. 1991. Computationally feasible bounds for partially observed Markov decision processes. *Oper. Res.* 39(1):162—175
52. Yu H, Bertsekas D. 2004. *Discretized approximations for POMDP with average cost*. In *Proc. UAI'04*, pp. 619–627. AUAI Press
53. Bork A, Junges S, Katoen JP, Quatmann T. 2020. *Verification of indefinite-horizon POMDPs*. In *Proc. ATVA'20*, vol. 12302 of *LNCS*, pp. 288–304. Springer
54. Bouton M, Tumova J, Kochenderfer M. 2020. *Point-based methods for model checking in partially observable Markov decision processes*. In *Proc. AAAI'20*, vol. 34(06), pp. 10061–10068. AAAI Press
55. Chatterjee K, Chmelík M, Gupta R, Kanodia A. 2015. *Qualitative analysis of POMDPs with temporal logic specifications for robotics applications*. In *Proc. ICRA'15*, pp. 325–330. IEEE
56. Junges S, Jansen N, Wimmer R, Quatmann T, Winterer L, et al. 2018. *Finite-state controllers of POMDPs using parameter synthesis*. In *Proc. AUAI'18*, pp. 519–529
57. Winterer L, Junges S, Wimmer R, Jansen N, Topcu U, et al. 2021. Strategy synthesis for POMDPs in robot planning via game-based abstractions. *IEEE Trans. Autom. Control* 1040–1054(66):3
58. Carr S, Jansen N, Topcu U. 2020. *Verifiable RNN-based policies for POMDPs under temporal logic constraints*. In *Proc. IJCAI'20*, pp. 4121–4127
59. Giro S, Rabe M. 2012. *Verification of partial-information probabilistic systems using counterexample-guided refinements*. In *Proc. ATVA'12*, vol. 7561 of *LNCS*, pp. 333–348. Springer
60. Cerný P, Chatterjee K, Henzinger T, Radhakrishna A, Singh R. 2011. *Quantitative synthesis for concurrent programs*. In *Proc. CAV'11*, vol. 6806 of *LNCS*, pp. 243–259. Springer
61. Suilen M, Jansen N, Cubuktepe M, Topcu U. 2020. *Robust policy synthesis for uncertain POMDPs via convex optimization*. In *Proc. IJCAI'20*, pp. 4113–4120
62. Carr S, Jansen N, Wimmer R, Fu J, Topcu U. 2018. Human-in-the loop synthesis for partially observable Markov decision processes. *Proc. ACC'18* :762–769
63. Feng L, Wiltsche C, Humphrey L, Topcu U. 2016. Synthesis of human-in-the-loop control protocols for autonomous systems. *IEEE Trans. Autom. Sci. Eng.* 13(2):450–462
64. Alur R, Henzinger T, Kupferman O. 2002. Alternating-time temporal logic. *J. ACM* 49(5):672–713
65. Chen T, Forejt V, Kwiatkowska M, Parker D, Simaitis A. 2013. Automatic verification of competitive stochastic systems. *FMSD* 43(1):61–92
66. von Neumann J, Morgenstern O, Kuhn H, Rubinstein A. 1944. *Theory of Games and Economic*

Behavior. Princeton University Press
67. Martin D. 1998. The determinacy of Blackwell games. *J. Symbolic Logic* 63(4):1565–1581
68. Condon A. 1993. On algorithms for simple stochastic games. *Advances in computational complexity theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* 13:51–73
69. Kelmendi E, Krämer J, Kretínský J, Weininger M. 2018. *Value iteration for simple stochastic games: stopping criterion and learning algorithm*. In *Proc. CAV'18*, vol. 10981 of *LNCS*, pp. 623–642. Springer
70. Chatterjee K, Henzinger T. 2006. *Strategy improvement and randomized subexponential algorithms for stochastic parity games*. In *Proc. STACS'06*, vol. 3884 of *LNCS*, pp. 512–523. Springer
71. Chatterjee K, Henzinger T. 2012. A survey of stochastic ω-regular games. *J. CSS* 78(2):394–413
72. Chen T, Kwiatkowska M, Simaitis A, Wiltsche C. 2013. *Synthesis for multi-objective stochastic games: an application to autonomous urban driving*. In *Proc. QEST'13*, vol. 8054 of *LNCS*, pp. 322–337. Springer
73. Basset N, Kwiatkowska M, Wiltsche C. 2018. Compositional strategy synthesis for stochastic games with multiple objectives. *IC* 261(3):536–587
74. Basset N, Kwiatkowska M, Topcu U, Wiltsche C. 2015. *Strategy synthesis for stochastic games with multiple long-run objectives*. In *Proc. TACAS'15*, vol. 9035 of *LNCS*, pp. 256–271. Springer
75. Feng L, Wiltsche C, Humphrey L, Topcu U. 2015. *Controller synthesis for autonomous systems interacting with human operators*. In *Proc. ICCPS'15*, pp. 70–79. ACM
76. Junges S, Jansen N, Katoen JP, Topcu U, Zhang R. 2018. *Model Checking for Safe Navigation Among Humans*. In *Proc. QEST'18*, vol. 11024 of *LNCS*, pp. 207–222. Springer
77. Glazier T, Cámara J, Schmerl B, Garlan D. 2015. *Analyzing resilience properties of different topologies of collective adaptive systems*. In *Proc. SASOW'15*, pp. 55–60. IEEE
78. Glazier T, Garlan D, Schmerl B. 2020. *Automated management of Collections of autonomic systems*. In *Proc. ACSOS'20*, pp. 82–91. IEEE
79. Cámara J, Garlan D, Schmerl B, Pandey A. 2015. *Optimal planning for architecture-based self-adaptation via model checking of stochastic games*. In *Proc. SAC'15*, pp. 428–435. ACM
80. Chatterjee K, Henzinger T, Jobstmann B, Radhakrishna A. 2010. GIST*: a solver for probabilistic games*. In *Proc. CAV'10*, vol. 6174 of *LNCS*, pp. 665–669. Springer. pub.ist.ac.at/gist/
81. Cheng C, Knoll A, Luttenberger M, Buckl C. 2011. *GAVS+: an open platform for the research of algorithmic game solving*. In *Proc. TACAS'11*, vol. 6605 of *LNCS*, pp. 258–261. Springer. sourceforge.net/projects/gavsplus/
82. Nash J. 1950. Equilibrium points in n-person games. *Proc. Natl. Acad. Sci* 36:48–49
83. Kwiatkowska M, Norman G, Parker D, Santos G. 2021. Automatic verification of concurrent stochastic systems. *FMSD*
84. Osborne M, Rubinstein A. 2004. *An Introduction to Game Theory*. Oxford University Press
85. Bouyer P, Markey N, Stan D. 2014. *Mixed Nash equilibria in concurrent games*. In *Proc. FSTTCS'14*, vol. 29 of *LIPICS*, pp. 351–363
86. de Alfaro L, Henzinger T, Kupferman O. 2007. Concurrent reachability games. *TCS* 386(3):188–217
87. Chatterjee K, de Alfaro L, Henzinger T. 2013. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *J. CSS* 79(5):640–657
88. Chen T, Forejt V, Kwiatkowska M, Simaitis A, Wiltsche C. 2013. *On stochastic games with multiple objectives*. In *Proc. MFCS'13*, vol. 8087 of *LNCS*, pp. 266–277. Springer
89. Karmarkar N. 1984. A new polynomial-time algorithm for linear programming. *Combinatorica* 4(4):373–395
90. Papadimitriou C. 1994. On the complexity of the parity argument and other inefficient proofs

of existence. *J. CSS* 48(3):498–532
91. Hansen K, Ibsen-Jensen R, Miltersen P. 2011. The complexity of solving reachability games using value and strategy iteration. *Theory Comput. Syst.* 55:380–403
92. Kwiatkowska M, Norman G, Parker D, Santos G. 2020. *Multi-player equilibria verification for concurrent stochastic games*. In *Proc. QEST'20*, LNCS, pp. 74–95. Springer
93. Daskalakis C, Goldberg P, Papadimitriou C. 2009. The complexity of computing a Nash equilibrium. *Com. ACM* 52(2):89–97
94. Chatterjee K, Majumdar R, Jurdziński M. 2004. *On Nash equilibria in stochastic games*. In *Proc. CSL'04*, vol. 3210 of *LNCS*, pp. 26–40. Springer
95. Ummels M. 2010. Stochastic multiplayer games: Theory and algorithms. Ph.D. thesis, RWTH Aachen University
96. Brihaye T, Bruyère V, Goeminne A, Raskin JF, van den Bogaard M. 2019. *The complexity of subgame perfect equilibria in quantitative reachability games*. In *Proc. CONCUR'19*, vol. 140 of *LIPICS*, pp. 13:1–13:16
97. Gutierrez J, Najib M, Giuseppe P, Wooldridge M. 2019. *Equilibrium design for concurrent games*. In *Proc. CONCUR'19*, vol. 140 of *LIPICS*, pp. 22:1–22:16
98. Bouyer P, Markey N, Stan D. 2016. *Stochastic equilibria under imprecise deviations in terminal-reward concurrent games*. In *Proc. GandALF'16*, vol. 226 of *EPTCS*, pp. 61–75
99. Jensen H. 1996. *Model checking probabilistic real time systems*. In *Proc. Nordic Workshop Programming Theory*, pp. 247–261
100. Kwiatkowska M, Norman G, Segala R, Sproston J. 2002. Automatic verification of real-time systems with discrete probability distributions. *TCS* 282:101–150
101. Beauquier D. 2003. Probabilistic timed automata. *TCS* 292(1):65–84
102. Kwiatkowska M, Norman G, Parker D, Sproston J. 2006. Performance analysis of probabilistic timed automata using digital clocks. *FMSD* 29:33–78
103. Kwiatkowska M, Norman G, Parker D. 2019. *Verification and control of turn-based probabilistic real-time games*. In *The Art of Modelling Computational Systems*, vol. 11760 of *LNCS*, pp. 379–396. Springer
104. Forejt V, Kwiatkowska M, Norman G, Trivedi A. 2016. Expected reachability-time games. *TCS* 631:139–160
105. Jurdziński M, Kwiatkowska M, Norman G, Trivedi A. 2009. *Concavely-priced probabilistic timed automata*. In *Proc. CONCUR'09*, vol. 5710 of *LNCS*, pp. 415–430. Springer
106. Kwiatkowska M, Norman G, Sproston J, Wang F. 2007. Symbolic model checking for probabilistic timed automata. *IC* 205(7):1027–1077
107. Jovanovic A, Kwiatkowska M, Norman G, Peyras Q. 2017. Symbolic optimal expected time reachability computation and controller synthesis for probabilistic timed automata. *TCS* 669:1–21
108. Kwiatkowska M, Norman G, Parker D. 2009. *Stochastic games for verification of probabilistic timed automata*. In *Proc. FORMATS'09*, vol. 5813 of *LNCS*, pp. 212–227. Springer
109. Tkachev I, Abate A. 2013. *Formula-free finite abstractions for linear temporal verification of stochastic hybrid systems*. In *Proc. HSCC'13*, pp. 283–292
110. Haesaert S, Soudjani S, Abate A. 2018. *Temporal logic control of general Markov decision processes by approximate policy refinement*. In *Proc. ADHS'18*, pp. 73–78
111. Clarke E, Grumberg O, Jha S, Lu Y, Veith H. 2000. *Counterexample-guided abstraction refinement*. In *Proc. CAV'00*, vol. 1855 of *LNCS*, pp. 154–169. Springer
112. D'Argenio P, Jeannet B, Jensen H, Larsen K. 2001. *Reachability analysis of probabilistic systems by successive refinements*. In *Proc. PAPM/PROBMIV'01*, vol. 2165 of *LNCS*, pp. 39–56. Springer
113. D'Argenio P, Jeannet B, Jensen H, Larsen K. 2002. *Reduction and refinement strategies for probabilistic analysis*. In *Proc. PAPM/PROBMIV'02*, vol. 2399 of *LNCS*, pp. 57–76. Springer
114. Segala R, Lynch N. 1995. Probabilistic simulations for probabilistic processes. *Nordic J. Com-*

puting 2(2):250–273
115. Wachter B, Zhang L, Hermanns H. 2007. *Probabilistic model checking modulo theories*. In *Proc. QEST'07*, pp. 129–140. IEEE
116. Hermanns H, Wachter B, Zhang L. 2008. *Probabilistic CEGAR*. In *Proc. CAV'08*, vol. 5123 of *LNCS*, pp. 162–175. Springer
117. Graf S, Saidi H. 1997. *Construction of abstract state graphs with PVS*. In *Proc. CAV'97*, vol. 1254 of *LNCS*, pp. 72–83. Springer
118. Han T, Katoen JP, Damman B. 209. Counterexample generation in probabilistic model checking. *IEEE Trans. Softw. Eng.* 35(2):241–257
119. Hahn E, Hermanns H, Wachter B, Zhang L. 2010. *PASS: abstraction refinement for infinite probabilistic models*. In *Proc. TACAS'10*, vol. 6105 of *LNCS*, pp. 353–357. Springer. depend.cs.uni-saarland.de/tools/pass/
120. Kattenbelt M, Kwiatkowska M, Norman G, Parker D. 2010. A game-based abstraction-refinement framework for Markov decision processes. *FMSD* 36(3):246–280
121. Kwiatkowska M, Norman G, Parker D, Qu H. 2013. Compositional probabilistic verification through multi-objective model checking. *IC* 232:38–65
122. Basset N, Kwiatkowska M, Wiltsche C. 2014. *Compositional controller synthesis for stochastic games*. In *Proc CONCUR'14*, vol. 8704 of *LNCS*, pp. 173–187. Springer
123. Chatterjee K, Doyen L. 2014. Partial-observation stochastic games: How to win when belief fails. *ACM TOCL* 15(2):1–44
124. Simaan M, Cruz J. 1973. On the Stackelberg strategy in nonzero-sum games. *J. Optim. Theory. Appl.* 533–555(11):5
125. Aumann R. 1974. Subjectivity and correlation in randomized strategies. *J. Math. Econ.* 1(2):67–96
126. Battigalli P, Dufwenberg M. 2009. Dynamic psychological games. *J. Econ. Theory* 244:67–96
127. Kaelbling L, Littman M, Moore A. 1996. Reinforcement learning: a survey. *J. Artif. Intell. Res.* 4:237–285
128. d'Avila Garcez A, Lamb L, Gabbay D. 2009. *Neural-Symbolic Cognitive Reasoning*. Cognitive Technologies. Springer