Formal Methods in System Design manuscript No.
(will be inserted by the editor)

Automatic Verification of Competitive Stochastic Systems

**Taolue Chen · Vojtěch Forejt ·
Marta Kwiatkowska · David Parker ·
Aistis Simaitis**

Received: date / Accepted: date

Abstract We present automatic verification techniques for the modelling and analysis of probabilistic systems that incorporate competitive behaviour. These systems are modelled as turn-based stochastic multi-player games, in which the players can either collaborate or compete in order to achieve a particular goal. We define a temporal logic called rPATL for expressing quantitative properties of stochastic multi-player games. This logic allows us to reason about the collective ability of a set of players to achieve a goal relating to the probability of an event's occurrence or the expected amount of cost/reward accumulated. We give an algorithm for verifying properties expressed in this logic and implement the techniques in a probabilistic model checker, as an extension of the PRISM tool. We demonstrate the applicability and efficiency of our methods by deploying them to analyse and detect potential weaknesses in a variety of large case studies, including algorithms for energy management in Microgrids and collective decision making for autonomous systems.

Keywords Quantitative verification · Probabilistic model checking · Stochastic multi-player games · Probabilistic temporal logic

Taolue Chen · Vojtěch Forejt · Marta Kwiatkowska · Aistis Simaitis
Department of Computer Science, University of Oxford, Oxford, United Kingdom
E-mail: firstname.lastname@cs.ox.ac.uk

David Parker
School of Computer Science, University of Birmingham, Birmingham, United Kingdom
E-mail: d.a.parker@cs.bham.ac.uk

1 Introduction

Automatic verification techniques for probabilistic systems have been successfully applied in a variety of fields, from wireless communication protocols to dynamic power management schemes to quantum cryptography. These systems are inherently stochastic, because of, for example, unreliable communication media, faulty components or the use of randomisation. Automatic techniques such as *probabilistic model checking* provide a means to model and analyse these systems against a range of quantitative properties. In particular, when systems also exhibit *nondeterministic* behaviour, for example due to concurrency, underspecification or control, the subtle interplay between the probabilistic and nondeterministic aspects of the system often makes a manual analysis difficult and error-prone.

When modelling *open* systems, the designer also has to account for the behaviour of components it does not control, and which could have differing or opposing goals, giving rise to *competitive* behaviour. This occurs in many cases, such as algorithms and protocols for distributed consensus, energy management, sensor network co-ordination or security. In such situations, it is natural to adopt a *game-theoretic* view, modelling a system as a game between different players. Automatic verification has been successfully deployed in this context, e.g. in the analysis of security [27] or communication protocols [36].

In this paper, we present an extensive framework for modelling and automatic verification of systems with both probabilistic *and* competitive behaviour, using *stochastic multi-player games* (SMGs). We introduce a probabilistic alternating-time temporal logic with rewards (rPATL) for expressing quantitative properties of this model and develop model checking algorithms for it. We then build a probabilistic model checker, based on the PRISM tool [28], which provides a high-level language for modelling SMGs and implements rPATL model checking for their analysis. Finally, to illustrate the applicability of our framework, we develop several large case studies in which we identify potential weaknesses and unexpected behaviour that would have been difficult to find with existing probabilistic verification techniques.

We model competitive stochastic systems as *turn-based* SMGs, where, in each state of the model, one player chooses between several actions, the outcome of which can be probabilistic. Turn-based games are a natural way to model many real-life applications. One example is when modelling several components executing concurrently under the control of a particular (e.g. round-robin, randomised) scheduler; in this case, nondeterminism in the model arises due to the choices made by each individual component. Another example is when we choose to explicitly model the (possibly unknown) scheduling of components as one player and the choices of components as other players.

The logic rPATL is an extension of the logic PATL [18], which is itself a probabilistic extension of ATL [2], a widely used logic for reasoning about multi-player games and multi-agent systems. rPATL allows us to state that a *coalition* of players has a *strategy* which can ensure that either the *probability* of an event's occurrence or an *expected reward* measure meets some threshold, e.g. for a network protocol: "can nodes 1 and 2 collaborate so that the probability of the protocol terminating within 45 seconds is at least 0.95, whatever nodes 3 and 4 do?"

We place particular emphasis on *reward* (or, equivalently, *cost*) related measures. This allows us to reason quantitatively about a system's use of resources, such as time spent or energy consumed; or, we can use rewards as an algorithm design mechanism to validate, benchmark or synthesise strategies for components by rewarding or penalising them for certain behaviour. rPATL can state, for example, "can sensor 1 ensure that the expected energy used, *if the algorithm terminates*, is less than $75mJ$, for any actions of sensors 2, 3, and 4?". To the best of our knowledge, this is the first logic able to express such properties.

We include in rPATL three different *cumulative expected reward* operators. Cumulative properties naturally capture many useful system properties, as has been demonstrated for verification of other types of probabilistic models [24], and as proves to be true for the systems we investigate. Indicative examples from our case studies are "the maximum expected execution cost of a task in a Microgrid" and "the minimum expected number of messages required to reach a consensus". Several other reward-based objectives exist that we do not explicitly consider, including discounted rewards (useful, e.g., in economics, but less so for the kind of systems we target) and long-run average reward (also useful, but practical implementations become complex in stochastic games [23]). We also mention that discounted reward objectives could be expressed within our framework by modifying the underlying game.

We present model checking algorithms for rPATL. A practical advantage of the logic is that, like for ATL, model checking reduces to analysing *zero-sum two-player* games. rPATL properties referring to the probability of an event are checked by solving simple stochastic two-player games, for which efficient techniques exist [19, 23]. We also describe model checking for the extended logic, rPATL*, again by reducing to known methods for two-player games [15]. For the reward operators of rPATL, we devise new algorithms.

Lastly, we implement our model checking techniques in a prototype tool and then develop and analyse several large case studies. In particular, we study algorithms for smart energy management [26] and distributed consensus in a sensor network [33]. In the first case, we use our techniques to reveal a weakness in the algorithm: we show that users may have a high incentive to deviate from the original algorithm, and propose modifications to solve the problem. For the consensus algorithm, we identify unexpected trade-offs in the performance of the algorithm when using our techniques to evaluate possible strategies for sensors.

Contributions. In summary, the contributions of this paper are:

- A comprehensive *framework* for analysis of competitive stochastic systems;
- A *logic* rPATL for specifying quantitative properties of stochastic multi-player games including, in particular, novel operators for *costs and rewards*, and their *model checking algorithms*;
- Implementation of a *tool* for modelling and rPATL model checking of SMGs;
- Development and analysis of several large new *case studies.*

This paper is an extended version of [16], which adds, in particular, the extended logic rPATL*, reward-bounded properties and operators for deciding the existence of price-bounded coalitions to satisfy an rPATL formula. We also provide proofs for the results and algorithms in the paper.

Paper structure. The remainder of the paper is structured as follows. In the next section, we review related work in this area. Then, Section 2 provides some background material on stochastic multi-player games. Section 3 introduces the logic rPATL and its extensions. The corresponding model checking algorithms are presented in Section 4. Section 5 describes an implementation of our model checking framework and its application to several large case studies. Proofs for the main results in the paper are contained in the appendix.

1.1 Related work

There are various theoretical results relating to probabilistic temporal logics in a game-theoretic setting [18,37,38,1,10,34,5] but, to our knowledge, this is the first work to consider a practical implementation, as well as modelling and automated verification of case studies. The logics PATL and PATL*, which we extend in this paper, were first introduced in [18], where the complexity of model checking for these logics was studied using a reduction to probabilistic parity games. In [37], simulation relations for stochastic games are proposed and shown to preserve fragments of PATL; an algorithm for computing such simulations is given in [38]. In [1], the logic PATL* is used in a theoretical framework for analysing security protocols. Various extensions of PATL have also been considered: [10] studies alternative semantics for the logic, using the notion of prediction denotation functions; [34] gives decidability and complexity results for a more expressive logic that incorporates partial information; and [5] presents (un)decidability results for another richer logic, with emphasis on the subtleties of nested properties. We note that all of the above, except [5], use concurrent, rather than turn-based, games and none consider reward properties. Turn-based games can be viewed as a special case of concurrent games where only one player has a choice of actions in every state.

There has been much research on algorithms to solve stochastic games, e.g. [19, 23,15,12,35], but these do not consider a modelling framework, implementation or case studies. Moreover, the reward-based properties that we introduce in this paper have not been studied in depth. Probabilistic model checking for a multi-agent system (a negotiation protocol) is considered in [7], using PRISM [28], but this is done by fixing a particular probabilistic strategy and analysing a Markov chain rather than a stochastic game. In [32], a quantitative generalisation of the μ-calculus is proposed, and shown to be able to encode stochastic parity games. The techniques are illustrated using a model of futures market investor, to which we will apply our techniques in Section 5. We will also analyse a model of a team formation protocol taken from [17], which performs probabilistic model checking using Markov chains and Markov decisions processes, as well as analysing simple properties of stochastic two-player games.

Stochastic games are also useful for *synthesis*, as in, for example, [11], which synthesises concurrent programs operating under randomised schedulers. Related to this is the tool GIST [14], a stochastic game solver with support for *qualitative* ω-regular objectives via a reduction to (non-probabilistic) two-player games. This is targeted specifically at synthesis problems, rather than general modelling and verification of competitive systems, as we do here. Finally, we also mention the tools MCMAS [30] and MOCHA [3], which are powerful model checkers for *non-probabilistic* multi-agent systems. In addition to the logic ATL, the former includes support for epistemic operators that reason about the knowledge of agents, a direction that we do not consider in this paper.

2 Preliminaries

We begin with some background on stochastic multi-player games. For a finite set X, we denote by $\mathcal{D}(X)$ the set of discrete probability distributions over X.

Definition 1 (SMG) A (turn-based) *stochastic multi-player game* (SMG) is a tuple $\mathcal{G} = \langle \Pi, S, A, (S_i)_{i\in\Pi}, \Delta, AP, \chi\rangle$, where: Π is a finite set of *players*; S is a finite, non-empty set of *states*; A is a finite, non-empty set of *actions*; $(S_i)_{i\in\Pi}$ is a partition of S; $\Delta : S \times A \to \mathcal{D}(S)$ is a (partial) *transition function*; AP is a finite set of *atomic propositions*; and $\chi : S \to 2^{AP}$ is a *labelling function.*

In each state $s \in S$ of the SMG $\mathcal{G}$, the set of *available* actions is denoted by $A(s) \overset{\text{def}}{=} \{a \in A \mid \Delta(s, a){\neq}\bot\}$. We assume that $A(s) \neq \emptyset$ for all s. The choice of action to take in s is under the control of exactly one player, namely the player $i \in \Pi$ for which $s \in S_i$. Once action $a \in A(s)$ is selected, the successor state is chosen according to the probability distribution $\Delta(s, a)$. A *path* of $\mathcal{G}$ is a possibly infinite sequence $\lambda = s_0a_0s_1a_1\ldots$ such that $a_j \in A(s_j)$ and $\Delta(s_j, a_j)(s_{j+1}) > 0$ for all j. We use st_λ to denote $s_0s_1\ldots$, and $st_\lambda(j)$ for s_j. Also, by λ_i we denote the suffix of the path λ starting in state s_i, e.g. $\lambda_1 = s_1a_1s_2\cdots$. The set of all infinite paths in $\mathcal{G}$ is $\Omega_\mathcal{G}$ and the set of infinite paths starting in state s is $\Omega_{\mathcal{G},s}$. Similarly, we use $\Omega^+_\mathcal{G}$ and $\Omega^+_{\mathcal{G},s}$ to denote the sets of finite paths.

A *strategy* for player $i \in \Pi$ in $\mathcal{G}$ is a function $\sigma_i : (SA)^*S_i \to \mathcal{D}(A)$ which, for each path $\lambda{\cdot}s \in \Omega^+_\mathcal{G}$ where $s \in S_i$, selects a probability distribution $\sigma_i(\lambda{\cdot}s)$ over $A(s)$. The set of all strategies for player i is denoted Σ_i. A strategy σ_i is called *memoryless* if $\sigma_i(\lambda{\cdot}s) = \sigma_i(\lambda'{\cdot}s)$ for all paths $\lambda{\cdot}s, \lambda'{\cdot}s \in \Omega^+_\mathcal{G}$, and *deterministic* if $\sigma_i(\lambda{\cdot}s)$ is a Dirac distribution for all $\lambda{\cdot}s \in \Omega^+_\mathcal{G}$. A *strategy profile* $\sigma = \sigma_1, \ldots, \sigma_{|\Pi|}$ comprises a strategy for all players in the game. Under a strategy profile σ, the behaviour of $\mathcal{G}$ is fully probabilistic and we define a *probability measure* $\mathrm{Pr}^\sigma_{\mathcal{G},s}$ over the set of all paths $\Omega_{\mathcal{G},s}$ in standard fashion (see, e.g. [12]). Given a random variable $X : \Omega_{\mathcal{G},s} \to \mathbb{R}$, we define the *expected value* of X to be $\mathbb{E}^\sigma_{\mathcal{G},s}[X] \overset{\text{def}}{=} \int_{\Omega_{\mathcal{G},s}} X \, d\mathrm{Pr}^\sigma_{\mathcal{G},s}$.

We also augment games with *reward structures* $r : S \to \mathbb{Q}_{\geq 0}$, which are labelling functions mapping each state to a non-negative rational reward. To simplify presentation, we only use state rewards, but note that transition/action rewards can easily be encoded by adding an auxiliary state per transition/action to the model.

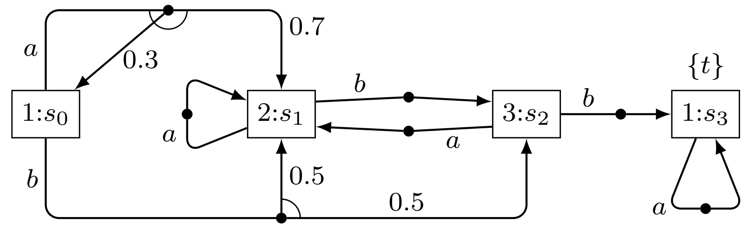


Fig. 1: Example SMG.

Example 1 Consider the SMG shown in Figure 1, with $\Pi{=}\{1, 2, 3\}$. The player i controlling a state s is shown as $i{:}s$ in the figure, i.e. $S_1{=}\{s_0, s_3\}$, $S_2{=}\{s_1\}$,

$S_3{=}\{s_2\}$. We have actions $A{=}\{a,b\}$ and, e.g., $\Delta(s_0,a)(s_1){=}0.7$. State s_3 is labelled with atomic proposition t, and all others are labelled with no atomic propositions. An example of an infinite path is $\lambda = s_0as_0bs_1bs_2bs_3(as_3)^\omega$; if player actions were chosen without the use of randomisation, the measure of this path would be 0.15 (i.e. $0.3 \cdot 0.5$).

3 Property Specification: The Logic rPATL

3.1 rPATL

We now present a temporal logic called rPATL (Probabilistic Alternating-time Temporal Logic with Rewards) for expressing quantitative properties of SMGs. Throughout the section, we assume a fixed SMG $\mathcal{G} = \langle \Pi, S, A, (S_i)_{i\in\Pi}, \Delta, AP, \chi\rangle$.

Definition 2 (rPATL) The syntax of rPATL is given by the grammar:

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi \wedge \phi \mid \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle \mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{\star}\phi]$$
$$\psi ::= \mathsf{X}\,\phi \mid \phi\,\mathsf{U}^{\leq k}\,\phi \mid \phi\,\mathsf{U}\,\phi$$

where $a \in AP$, $C \subseteq \Pi$, $\bowtie \in \{<,\leq,\geq,>\}$, $q \in \mathbb{Q} \cap [0,1]$, $x \in \mathbb{Q}_{\geq 0}$, r is a reward structure, $\star \in \{0,\infty,c\}$ and $k \in \mathbb{N}$.

rPATL is a CTL-style branching-time temporal logic, where we distinguish between state formulae (ϕ) and path formulae (ψ). We adopt the coalition operator $\langle\langle C\rangle\rangle$ of ATL [2], combining it with the probabilistic operator $\mathsf{P}_{\bowtie q}$ and path formulae from PCTL [25,8] and a generalisation of the reward operator $\mathsf{R}^{r}_{\bowtie x}$ from [24].

An example of typical usage of the coalition operator is $\langle\langle\{1,2\}\rangle\rangle\mathsf{P}_{\geq 0.5}[\psi]$, which means "players 1 and 2 have a strategy to ensure that the probability of path formula ψ being satisfied is at least 0.5, regardless of the strategies of other players". As path formulae, we allow the standard temporal operators X ("next"), $\mathsf{U}^{\leq k}$ ("bounded until") and U ("until"). As usual, we can derive other common temporal operators such as $\mathsf{F}\,\phi \equiv \top\,\mathsf{U}\,\phi$ ("eventually").

3.2 Rewards

Before presenting the semantics of rPATL, we discuss the reward operators in the logic. We focus on the *expected cumulative reward* to reach a target, i.e. the expected sum of rewards cumulated along a path until a state from a specified set $T \subseteq S$ is reached. To cope with the variety of different properties encountered in practice, we introduce three variants, which differ in the way they handle the case where T is *not* reached. The three types are denoted by the parameter $\star$, one of 0, ∞ or c. These indicate that, when T is not reached, the reward is zero, infinite or equal to the cumulated reward along the whole path, respectively.

The motivation for selecting these particular types of rewards stems from our experience of devising rPATL specifications for practical case studies (which we present later in Section 5). Each reward type is applicable in different situations. If our goal is, for example, to minimise the expected time for algorithm completion, then it is natural to assume a value of infinity upon non-completion ($\star$=∞). Consider, on the other hand, the case where we try to optimise a distributed algorithm by designing a reward structure that incentivises certain kinds of behaviour and then maximising it over the lifetime of the algorithm's execution. In this case, we might opt for type $\star$=0 to avoid favouring situations where the algorithm does not terminate. In other cases, e.g. when modelling algorithm's resource consumption, we might prefer to use type $\star = c$, to compute resources used regardless of termination.

We formalise these notions of rewards by defining *reward functions* that map each possible path in the game $\mathcal{G}$ to a cumulative reward value.

Definition 3 (Reward Function) For an SMG $\mathcal{G}$, a reward structure r, type $\star \in \{0, \infty, c\}$ and a set $T \subseteq S$ of target states, the *reward function* $\mathit{rew}(r, \star, T) : \Omega_{\mathcal{G}} \to \mathbb{R}_{\geq 0}$ is a random variable defined as follows. For $\lambda \in \Omega_{\mathcal{G}}$:

$$\mathit{rew}(r,\star,T)(\lambda) \overset{\text{def}}{=} \begin{cases} g(\star) & \text{if } \forall j \in \mathbb{N} : \mathit{st}_\lambda(j) \notin T, \\ \sum_{j=0}^{k-1} r(\mathit{st}_\lambda(j)) & \text{otherwise, where } k = \min\{j \mid \mathit{st}_\lambda(j) \in T\}, \end{cases}$$

and where $g(\star) = \star$ if $\star \in \{0, \infty\}$ and $g(\star) = \sum_{j \in \mathbb{N}} r(\mathit{st}_\lambda(j))$ if $\star = c$. The *expected reward* from a state $s \in S$ of $\mathcal{G}$ under a strategy profile σ is the expected value of the reward function, $\mathbb{E}^{\sigma}_{\mathcal{G},s}[\mathit{rew}(r, \star, T)]$.

3.3 Semantics of rPATL

Now, we define the semantics of rPATL. Formulae are interpreted over states of a game $\mathcal{G}$; we write $s \models \phi$ to indicate that state s of $\mathcal{G}$ satisfies the formula ϕ and define $\mathit{Sat}(\phi) \overset{\text{def}}{=} \{s \in S \mid s \models \phi\}$ as the set of states satisfying ϕ. The meaning of atomic propositions and logical connectives is standard. For the $\langle\langle C \rangle\rangle \mathsf{P}_{\bowtie q}$ and $\langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}$ operators, we give the semantics via a reduction to a two-player game called a *coalition game*.

Definition 4 (Coalition Game) For a *coalition* of players $C \subseteq \Pi$ of SMG $\mathcal{G}$, we define the *coalition game* of $\mathcal{G}$ induced by C as the stochastic two-player game $\mathcal{G}_C = \langle \{1, 2\}, S, A, (S'_1, S'_2), \Delta, AP, \chi \rangle$ where $S'_1 = \cup_{i \in C} S_i$ and $S'_2 = \cup_{i \in \Pi \setminus C} S_i$.

Definition 5 (rPATL Semantics) The *satisfaction relation* $\models$ for rPATL is defined inductively for each state s of $\mathcal{G}$, as follows:

$$\begin{array}{lcl}
s \models \top & & \text{always} \\
s \models a & \Leftrightarrow & a \in \chi(s) \\
s \models \neg\phi & \Leftrightarrow & s \not\models \phi \\
s \models \phi_1 \wedge \phi_2 & \Leftrightarrow & s \models \phi_1 \text{ and } s \models \phi_2 \\
s \models \langle\langle C \rangle\rangle \mathsf{P}_{\bowtie q}[\psi] & \Leftrightarrow & \text{In coalition game } \mathcal{G}_C,\ \exists \sigma_1 \in \Sigma_1 \text{ such that } \forall \sigma_2 \in \Sigma_2 \\
 & & \mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi) \bowtie q \\
s \models \langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^\star \phi] & \Leftrightarrow & \text{In coalition game } \mathcal{G}_C,\ \exists \sigma_1 \in \Sigma_1 \text{ such that } \forall \sigma_2 \in \Sigma_2 \\
 & & \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r, \star, \mathit{Sat}(\phi))] \bowtie x
\end{array}$$

where $\mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi) \overset{\text{def}}{=} \mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\{\lambda \in \Omega_{\mathcal{G}_C,s} \mid \lambda \models \psi\})$ and for any path λ in $\Omega_{\mathcal{G}}$:

$$\begin{array}{lcl}
\lambda \models \mathsf{X}\, \phi & \Leftrightarrow & \mathit{st}_\lambda(1) \models \phi \\
\lambda \models \phi_1 \,\mathsf{U}^{\leq k}\, \phi_2 & \Leftrightarrow & \mathit{st}_\lambda(i) \models \phi_2 \text{ for some } i \leq k \text{ and } \mathit{st}_\lambda(j) \models \phi_1 \text{ for } 0 \leq j < i \\
\lambda \models \phi_1 \,\mathsf{U}\, \phi_2 & \Leftrightarrow & \lambda \models \phi_1 \,\mathsf{U}^{\leq k}\, \phi_2 \text{ for some } k \in \mathbb{N}.
\end{array}$$

Example 2 Consider again the example SMG in Figure 1. The rPATL formula $\langle\langle\{1,3\}\rangle\rangle\mathsf{P}_{\geq 0.5}[\mathsf{F}\, t]$ is satisfied in states s_0, s_2 and s_3, i.e. players 1 and 3 can ensure that the probability of reaching t is at least 0.5, by both taking action b in their states. For example, from s_0, the measure of path $s_0 b s_2 b s_3 (a s_3)^\omega$ is 0.5, achieving the required bound. The formula $\langle\langle\{1,2\}\rangle\rangle\mathsf{P}_{\geq 0.5}[\mathsf{F}\, t]$, on the other hand, is satisfied only in s_3 because player 3 can always take action a in state s_2 ensuring that the t-labelled state s_3 is never reached from any other state.

Let r be a reward structure that assigns i to state s_i, i.e. $r(s_0)=0$, $r(s_1)=1$, $r(s_2)=2$ and $r(s_3)=3$. rPATL formula $\langle\langle\{1,3\}\rangle\rangle\mathsf{R}^r_{\leq 2}[\mathsf{F}^\infty t]$ is true in states s_2 (player 3 taking action b, ensuring the expected cumulated reward is exactly 2) and s_3 (expected reward is 0 by definition of F^∞). Formula $\langle\langle\{1\}\rangle\rangle\mathsf{R}^r_{\geq q}[\mathsf{F}^0 t]$ is false in all states for any $q>0$ because players 2 and 3, by both taking action a, can guarantee that s_3 is never reached from states other than s_3. However, $\langle\langle\emptyset\rangle\rangle\mathsf{R}^r_{\geq q}[\mathsf{F}^c t]$ is true in all states for any $q>0$, because any set of paths in the game which has positive measure also has infinite reward, i.e. the only path that has finite reward is $s_0(a s_0)^\omega$, but the measure of this path is 0.

3.4 Equivalences and Extensions

We can handle negated path formulae in a $\langle\langle C\rangle\rangle\mathsf{P}_{\bowtie q}$ operator by inverting the probability threshold, e.g.:

$$\langle\langle C\rangle\rangle\mathsf{P}_{\geq q}[\neg\psi] \equiv \langle\langle C\rangle\rangle\mathsf{P}_{\leq 1-q}[\psi].$$

This allows us to derive, for example, the G ("globally") and R ("release") operators, using the equivalences $\mathsf{G}\,\phi \equiv \neg(\mathsf{F}\,\neg\phi) \equiv \neg(\top\,\mathsf{U}\,\neg\phi)$ and $\phi_1\,\mathsf{R}\,\phi_2 \equiv \neg(\neg\phi_1\,\mathsf{U}\,\neg\phi_2)$. This is done in the same way as for PCTL [25,8] (but, interestingly, cannot be done for ATL, as shown in [29]).

In addition, from the determinacy result of [31] for zero-sum stochastic two-player games with Borel measurable payoffs, it follows that, e.g.:

$$\langle\langle C\rangle\rangle\mathsf{P}_{\geq q}[\psi] \equiv \neg\langle\langle \Pi\setminus C\rangle\rangle\mathsf{P}_{<q}[\psi]. \tag{1}$$

It is also useful to consider "quantitative" versions of the $\langle\langle C\rangle\rangle\mathsf{P}_{\bowtie q}$ and $\langle\langle C\rangle\rangle\mathsf{R}^r_{\bowtie x}$ operators, in the style of PRISM [28], which return numerical values. For the probabilistic operator, we have:

$$\begin{aligned}\langle\langle C\rangle\rangle\mathsf{P}_{\min=?}[\psi] &\overset{\text{def}}{=} \mathrm{Pr}^{\min,\max}_{\mathcal{G}_C,s}(\psi) \overset{\text{def}}{=} \inf_{\sigma_1\in\Sigma_1}\sup_{\sigma_2\in\Sigma_2}\mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi)\\ \langle\langle C\rangle\rangle\mathsf{P}_{\max=?}[\psi] &\overset{\text{def}}{=} \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\psi) \overset{\text{def}}{=} \sup_{\sigma_1\in\Sigma_1}\inf_{\sigma_2\in\Sigma_2}\mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\psi)\end{aligned}$$

and for the reward operator:

$$\begin{aligned}\langle\langle C\rangle\rangle\mathsf{R}^r_{\min=?}[\mathsf{F}^\star\phi] &\overset{\text{def}}{=} \mathbb{E}^{\min,\max}_{\mathcal{G}_C,s}[rew(r,\star,Sat(\phi))]\\ &\overset{\text{def}}{=} \inf_{\sigma_1\in\Sigma_1}\sup_{\sigma_2\in\Sigma_2}\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,\star,Sat(\phi))]\\ \langle\langle C\rangle\rangle\mathsf{R}^r_{\max=?}[\mathsf{F}^\star\phi] &\overset{\text{def}}{=} \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[rew(r,\star,Sat(\phi))]\\ &\overset{\text{def}}{=} \sup_{\sigma_1\in\Sigma_1}\inf_{\sigma_2\in\Sigma_2}\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,\star,Sat(\phi))]\,.\end{aligned}$$

3.5 Reward-bounded Properties and Price-bounded Coalitions

Next, we consider two extensions of rPATL which provide additional ways to reason about *bounded rewards* and *coalition prices.*

Reward-bounded properties. So far, rPATL's reward-based operators only allow us to reason about the existence of a coalition strategy to achieve a given bound on *expected* rewards. These do not, however, rule out the possibility of paths in the game that accumulate arbitrarily large (or even infinite) rewards, which may be undesirable when modelling resource consumption. Here, we take another view on the rewards by asking the question: does there exist a strategy for the coalition to achieve a given rPATL property using *bounded resources*? In particular, we extend the syntax of rPATL (see Definition 2) with the *reward-bounded until* temporal operator $\phi_1 \,\mathsf{U}^{r}_{\leq x} \phi_2$, where r is a reward structure, $x \in \mathbb{Q}_{\geq 0}$, and ϕ_1 and ϕ_2 are rPATL state formulae. Satisfaction by a path λ is defined as:

$$\lambda \models \phi_1 \,\mathsf{U}^{r}_{\leq x} \phi_2 \quad \Leftrightarrow \quad \text{there exists } k \geq 0 \text{ such that } st_\lambda(k) \models \phi_2,\ st_\lambda(j) \models \phi_1 \text{ for } 0 \leq j < k \text{ and } \textstyle\sum_{i=0}^{k} r(st_\lambda(i)) \leq x.$$

For example, rPATL formula $\langle\!\langle C \rangle\!\rangle \mathsf{P}_{>0.9}[\top \,\mathsf{U}^{r}_{\leq 100}\, \mathsf{success}]$ means that coalition C has a strategy to guarantee that, with probability greater than 0.9, a state satisfying $\mathsf{success}$ is reached whilst consuming no more than 100 units of reward r. We will describe how to model check such formulae in Section 4.4.

Price-bounded coalitions. Next, we consider more general queries about the existence of coalition strategies. In addition to asking whether there exists a strategy for a given coalition to ensure that a specified probability or reward bound is met, it is also natural to ask whether there exists a coalition of a certain size, for example, that has such a strategy. To formulate queries of this kind, we assume that each player $i \in \Pi$ has a prescribed non-negative *price* $p(i)$, and we define the price of a coalition $C \subseteq \Pi$ to be the sum of the prices of its players. We then extend the syntax of rPATL to allow the following *price-bounded* state formulae:

$$\langle\!\langle ? \rangle\!\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi] \quad \text{and} \quad \langle\!\langle ? \rangle\!\rangle_{\leq y} \mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{\star}\phi]$$

where $y \in \mathbb{Q}_{\geq 0}$ and the other parameters are as in Definition 2. Intuitively, these formulae are true if there is a coalition whose price is at most y and which can ensure that the probability or expected reward satisfies the given bound. Formally, for a state $s \in S$, the semantics is defined as:

$$\begin{aligned} s \models \langle\!\langle ? \rangle\!\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi] \quad &\Leftrightarrow \quad \exists C : \textstyle\sum_{i \in C} p(i) \leq y \text{ and } s \models \langle\!\langle C \rangle\!\rangle \mathsf{P}_{\bowtie q}[\psi] \\ s \models \langle\!\langle ? \rangle\!\rangle_{\leq y} \mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{\star}\phi] \quad &\Leftrightarrow \quad \exists C : \textstyle\sum_{i \in C} p(i) \leq y \text{ and } s \models \langle\!\langle C \rangle\!\rangle \mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{\star}\phi] \end{aligned}$$

We will consider model checking for these formulae in Section 4.4.

3.6 rPATL*

Finally, in this section, we discuss the logic rPATL*, which extends rPATL in the same way that PCTL* extends PCTL [8]. In particular, this allows LTL formulae to be provided as path formulae within the $\langle\langle C\rangle\rangle\mathsf{P}$ operator. The syntax of rPATL* is given by the following grammar:

$$\phi ::= \top \mid a \mid \neg\phi \mid \phi\wedge\phi \mid \langle\langle C\rangle\rangle\mathsf{P}_{\bowtie q}[\psi] \mid \langle\langle C\rangle\rangle\mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{\star}\phi]$$
$$\psi ::= \phi \mid \neg\psi \mid \psi\wedge\psi \mid \mathsf{X}\,\psi \mid \psi\,\mathsf{U}\,\psi$$

where $a \in AP$, $C \subseteq \Pi$, $\bowtie\in \{<,\leq,\geq,>\}$, $q \in \mathbb{Q}\cap[0,1]$, $x \in \mathbb{Q}_{\geq 0}$, r is a reward structure and $\star \in \{0,\infty,c\}$. Note that, for simplicity, we omit here the bounded variant of the until operator.

The semantics for state formulae are the same as for rPATL. The semantics for path formulae are as follows. For path $\lambda \in \Omega_{\mathcal{G}}$:

$$\begin{array}{lll}
\lambda \models \phi & \Leftrightarrow & st_{\lambda}(0) \models \phi \\
\lambda \models \neg\psi & \Leftrightarrow & \lambda \not\models \psi \\
\lambda \models \psi_1 \wedge \psi_2 & \Leftrightarrow & \lambda \models \psi_1 \text{ and } \lambda \models \psi_2 \\
\lambda \models \mathsf{X}\,\psi & \Leftrightarrow & \lambda_1 \models \psi \\
\lambda \models \psi_1\,\mathsf{U}\,\psi_2 & \Leftrightarrow & \lambda_i \models \psi_2 \text{ for some } i \in \mathbb{N} \text{ and } \lambda_j \models \psi_1 \text{ for } 0 \leq j < i\,.
\end{array}$$

Examples of rPATL* formulae include:

- $\langle\langle\{1,3\}\rangle\rangle\mathsf{P}_{\geq 1}[\mathsf{GF}\,\mathit{recharge}]$ - players 1 and 3 have a strategy to make sure that a "*recharge*" state is visited infinitely often with probability 1;
- $\langle\langle\{4\}\rangle\rangle\mathsf{P}_{>0.5}[(\mathsf{G}\,\mathit{safe}) \wedge (\mathsf{FG}\,\mathit{success})]$ - player 4 has a strategy such that, with probability greater than 0.5, the system ends up and remains in "*success*" states, while only visiting "*safe*" states, for any strategies of the other players.

We will describe model checking for rPATL* in Section 4.5.

4 Model Checking for rPATL

We now discuss model checking for rPATL, the key part of which is computation of probabilities and expected rewards for stochastic two-player games. The complexity of the rPATL model checking problem can be stated as follows.

Theorem 1 **(a)** *Model checking an rPATL formula with no* $\langle\langle C\rangle\rangle\mathsf{R}^{r}_{\bowtie x}[\mathsf{F}^{0}\phi]$ *operator and where k for the temporal operator* $\mathsf{U}^{\leq k}$ *is given in unary is in* NP $\cap$ coNP.
(b) *Model checking an arbitrary rPATL formula is in* NEXP $\cap$ coNEXP.

We give a proof of Theorem 1 in Appendix C. Note that our problem is at least as hard as solving simple stochastic two-player games, which is known to be in NP $\cap$ coNP [19] and for which the existence of polynomial time algorithms is a long-standing open problem. Nevertheless, we present efficient and practically usable algorithms for model checking rPATL, in which computation of numerical values is done by evaluating fixpoints up to a desired level of convergence (in the style of well-known *value iteration* algorithms [19]). In fact, this is the usual approach taken in probabilistic verification tools for simpler classes of models such as Markov chains and Markov decision processes.

It is possible to establish some complexity results for value iteration. For example, it was shown in [19] that it can be decided precisely whether the values being computed exceed a given threshold after a sufficiently large number of iterations. However, the best known lower bound on the number of iterations is exponential in the size of the game. A detailed discussion of the number of iterations needed to obtain a given precision can also be found in [13]. In practice, in our implementation, we decide when to terminate fixpoint computations based on a simple test of numerical convergence. We discuss this in more detail in Section 5.1.

4.1 The Basic Model Checking Algorithm

The basic algorithm for model checking an rPATL formula ϕ on an SMG $\mathcal{G}$ proceeds as for other branching-time logics, determining the set $\mathit{Sat}(\phi)$ recursively. Furthermore, as can be seen from the semantics, computing this set for atomic propositions or logical connectives is trivial. Thus, we only consider the $\langle\langle C\rangle\rangle\mathsf{P}_{\bowtie q}$ and $\langle\langle C\rangle\rangle\mathsf{R}^r_{\bowtie x}$ operators. Model checking of these reduces to computation of *optimal* probabilities or expected rewards, respectively, on the coalition game $\mathcal{G}_C$. For example, if $\rhd \in \{\geq, >\}$, then:

$$\begin{aligned} s \models \langle\langle C\rangle\rangle\mathsf{P}_{\rhd q}[\psi] \;&\Leftrightarrow\; \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\psi) \rhd q \\ s \models \langle\langle C\rangle\rangle\mathsf{R}^r_{\rhd x}[\mathsf{F}^\star\phi] \;&\Leftrightarrow\; \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,\mathit{Sat}(\phi))] \rhd x. \end{aligned}$$

Analogously, for operators $\leq$ and $<$, we simply swap min and max in the above. The following sections describe how to compute probabilities ($\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\psi)$) and expected rewards ($\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,\mathit{Sat}(\phi))]$) for the coalition game $\mathcal{G}_C$.

4.2 Computing Probabilities

Below, we show how to compute the probabilities $\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\psi)$ where ψ is each of the temporal operators X, $\mathsf{U}^{\leq k}$ and U. We omit the dual case since, thanks to determinacy (see equation (1)), we have that $\mathrm{Pr}^{\min,\max}_{\mathcal{G}_C,s}(\psi) = \mathrm{Pr}^{\max,\min}_{\mathcal{G}_{\Pi\setminus C},s}(\psi)$. The following results follow in near identical fashion to the corresponding statements for Markov decision processes [6]. We let opt^s denote either max or min, depending on whether state s of $\mathcal{G}_C$ is controlled by player 1 or 2:

$$\mathrm{opt}^s = \begin{cases} \max \text{ if } s \in S_1 \\ \min \text{ if } s \in S_2. \end{cases}$$

Then, for the X operator and state $s \in S$:

$$\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{X}\,\phi) = \mathrm{opt}^s_{a\in A(s)} \textstyle\sum_{s'\in \mathit{Sat}(\phi)} \Delta(s,a)(s').$$

Probabilities for the $\mathsf{U}^{\leq k}$ operator can be computed recursively. We have that $\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1\,\mathsf{U}^{\leq k}\,\phi_2)$ is equal to: 1 if $s \in \mathit{Sat}(\phi_2)$; 0 if $s \notin (\mathit{Sat}(\phi_1) \cup \mathit{Sat}(\phi_2))$; 0 if $k{=}0$ and $s \in \mathit{Sat}(\phi_1)\backslash\mathit{Sat}(\phi_2)$; and otherwise:

$$\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1\,\mathsf{U}^{\leq k}\,\phi_2) = \mathrm{opt}^s_{a\in A(s)} \textstyle\sum_{s'\in S} \Delta(s,a)(s') \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s'}(\phi_1\,\mathsf{U}^{\leq k-1}\,\phi_2).$$

The unbounded case can be computed via *value iteration* [19], i.e. using:

$$\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1\,\mathsf{U}\,\phi_2) = \mathrm{lim}_{k\to\infty}\, \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1\,\mathsf{U}^{\leq k}\,\phi_2).$$

In practice, this computation is terminated with a suitable convergence check (see Section 5.1). In addition, we mention that, for the case $\mathsf{F}\,\phi \equiv \top\,\mathsf{U}\,\phi$, the computation can also be reduced to quadratic programming [23].

4.3 Computing Rewards

Now, we discuss computation for the expected reward operators. We remark that, although phrased here in terms of rPATL model checking for SMGs, the techniques presented have general applicability for stochastic two-player games.

We show how to compute the optimal values $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,\mathit{Sat}(\phi))]$ for all three types $\star \in \{0,\infty,c\}$. As above, we omit the dual case where max and min are swapped. In this section, we fix a coalition game $\mathcal{G}_C$, a reward structure r, and a target set $T = \mathit{Sat}(\phi)$. We first make the following modifications to $\mathcal{G}_C$:

- fresh atomic propositions t and a_{rew} are added to target and positive reward states: $AP := AP \cup \{t, a_{\mathrm{rew}}\}$, $\forall s \in T : \chi(s) := \chi(s) \cup \{t\}$ and $\forall s \in S\,.\, r(s) > 0 : \chi(s) := \chi(s) \cup \{a_{\mathrm{rew}}\}$;
- target states are made absorbing: $\forall s \in T : A(s) := \{a\}, \Delta(s,a)(s){=}1, r(s){=}0$.

Our algorithms, like the ones for similar properties on simpler models [6], rely on computing fixpoints of certain sets of equations. As in the previous section, we assume that this is done by *value iteration* with an appropriate convergence criterion. As above, we let opt^s denote max if $s \in S_1$ and min if $s \in S_2$.

An important observation here is that optimal expected rewards for $\star \in \{\infty, c\}$ can be achieved by memoryless, deterministic strategies (see Appendix A.2). For $\star = 0$, however, finite-memory strategies are needed (see Appendix A.1).

4.3.1 The case $\star = c$

First, we use the results of [22] to identify the states from which the expected reward is infinite:

$$I := \{s \in S \mid \exists \sigma_1{\in}\Sigma_1\ \forall \sigma_2{\in}\Sigma_2\ \mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}(a_{rew})) > 0\}$$

where $\mathit{inf}(a_{rew})$ is the set of all paths that visit a state satisfying a_{rew} infinitely often. The states in I are assigned infinite reward. To compute values for the remaining states, we remove the states in I from $\mathcal{G}_C$ before continuing. We then compute the *least* fixpoint of the following equations:

$$f(s) = \begin{cases} 0 & \text{if } s \in T \\ r(s) + \mathrm{opt}^s_{a \in A(s)} \sum_{s' \in S} \Delta(s,a)(s') \cdot f(s') & \text{otherwise} \end{cases} \tag{2}$$

and let $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,c,T)] = f(s)$. For a proof of correctness of the procedure, see Appendix B.1.

4.3.2 The case $\star = \infty$

Again, we start by identifying and removing states with infinite expected reward; in this case: $I := \{s \in S \mid s \models \langle\!\langle\{1\}\rangle\!\rangle \mathsf{P}_{<1}[\,\mathsf{F}\,t\,]\}$. Then, for all other states s, we compute the *greatest* fixpoint, *over* $\mathbb{R}$, of equations (2). The need for the greatest fixpoint arises because, in the presence of zero-reward cycles, multiple fixpoints may exist. For the previous case ($\star = c$), a least fixpoint of (2) gives the correct solution for such cases; here, the reward should be infinite. The computation is over $\mathbb{R}$ since, e.g., the function mapping all non-target states to ∞ may also be a fixpoint, which is not the one we are interested in.

To find the greatest fixpoint over $\mathbb{R}$, we first compute an over-approximation by changing all zero rewards to any $\varepsilon > 0$ and then evaluating the *least* fixpoint of (2) for the *modified* reward. Starting from the *new* initial values, value iteration now converges from above to the correct fixpoint (for a proof, see Appendix B.2). For the simpler case of Markov decision processes, an alternative approach based on removal of zero-reward end-components is possible [21], but this cannot be adapted efficiently to stochastic games.

4.3.3 The case $\star = 0$

As mentioned above, it does not suffice to consider memoryless strategies in this case. The optimal strategy may depend on the reward accumulated so far, which we denote as $\mathrm{r}(\lambda) \overset{\text{def}}{=} \sum_{s \in st_\lambda} r(s)$ for history λ. However, this is only needed until a certain reward bound B is reached, after which the optimal strategy picks actions that maximise the probability of reaching T (if multiple such actions exist, it picks the one with the highest expected reward). The bound B can be computed efficiently using algorithms for $\star{=}c$ and $\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\psi)$ and, in the worst case, can be exponential in the size of $\mathcal{G}$ and the reward structure r (see Appendix B.3).

For clarity, we assume that rewards are integers.[1] Let $R_{(s,k)}$ be the maximum expectation of $rew(r, 0, T)$ in state s after history λ with $\mathrm{r}(\lambda) = k$:

$$R_{(s,k)} \overset{\text{def}}{=} \max_{\sigma_1 \in \Sigma_1} \min_{\sigma_2 \in \Sigma_2} \Big[k \cdot \mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathsf{F}\, t) + \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,0,T)] \Big],$$

and $r_{\max} = \max_{s \in S} r(s)$. The algorithm works as follows:

1. Using the results of [22], identify the states that have infinite reward:

$$I := \{ s \in S \mid \exists \sigma_1{\in}\Sigma_1\ \forall \sigma_2{\in}\Sigma_2\ \mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}^t(a_{rew})) > 0 \}\}$$

   where $\mathit{inf}^t(a_{rew})$ is the set of all paths that visit a state satisfying $\mathsf{P}_{>0}[\mathsf{F}\, t] \wedge a_{rew}$ infinitely often. Then, assign infinite reward to states in I and remove them from the game.
2. For $B \leq k \leq B + r_{\max} - 1$ and for each state s:
   (a) Assign new reward $r'(s) = r(s) \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t)$;
   (b) Remove from $A(s)$ actions a that are sub-optimal for $\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t)$, i.e.:

$$\textstyle\sum_{s' \in S} \Delta(s,a)(s') \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s'}(\mathsf{F}\, t) < \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t);$$

   (c) Compute $R_{(s,k)}$ using the algorithm for $rew(r', c, T)$:

$$R_{(s,k)} = k \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t) + \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[rew(r', c, T)].$$

[1] Rational values can be handled by re-scaling all rewards by the lowest common multiple of the denominators of rewards appearing in the game. Note that re-scaling does not increase the size of the model, so the stated complexity results are not affected.

3. Find, for all $0 \leq k < B$ and states s, the *least* fixpoint of the equations:

$$R_{(s,k)} = \begin{cases} k & \text{if } s \in T \\ \text{opt}^s_{a \in A(s)} \sum_{s' \in S} \Delta(s,a)(s') \cdot R_{(s',k+r(s))} & \text{otherwise.} \end{cases}$$

4. The required values are then $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T)] = R_{(s,0)}$.

For a proof of correctness of the procedure, see Appendix B.3.

4.4 Reward-bounded Properties and Price-bounded Coalitions

Next, we consider the problem of model checking the *reward-* and *price-bounded* rPATL formulae presented in Section 3.5.

Reward bounded properties. In a similar fashion to the bounded until operator (see Section 4.2), probabilities for the $\phi_1 \,\mathsf{U}^r_{\leq x} \phi_2$ operator can be computed recursively. We have that $\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1 \,\mathsf{U}^r_{\leq x} \phi_2)$ is equal to: 1 if $s \in \mathit{Sat}(\phi_2)$; 0 if $s \notin (\mathit{Sat}(\phi_1) \cup \mathit{Sat}(\phi_2))$; 0 if $x < r(s)$ and $s \in \mathit{Sat}(\phi_1) \backslash \mathit{Sat}(\phi_2)$; and otherwise:

$$\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\phi_1 \,\mathsf{U}^r_{\leq x} \phi_2) = \text{opt}^s_{a \in A(s)} \sum_{s' \in S} \Delta(s,a)(s') \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s'}(\phi_1 \,\mathsf{U}^r_{\leq x - r(s)} \phi_2).$$

Note that model checking a reward-bounded until operator is equivalent to model-checking an unbounded until operator on an extended model that encodes the accummulated reward in its state. As in Section 4.3.3 above, we can assume that rewards in the model and the reward bound r are all integer values (to treat rational values, we re-scale by the lowest common multiple of their denominators). The state space of the extended model remains finite since rewards never need to exceed the bound r. A similar approach is taken in [4] for bounded-reward model-checking on the simpler model of discrete-time Markov chains.

Price-bounded coalitions. It follows from results for ATL [9] that the model checking problem for rPATL extended with formulae of the form $\langle\langle ? \rangle\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi]$ is NP-hard. We argue that the problem lies within the class P^{NP} of the polynomial hierarchy if the logic is extended with formulae $\langle\langle ? \rangle\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi]$ and $\langle\langle ? \rangle\rangle_{\leq y} \mathsf{R}^r_{\bowtie x}[\mathsf{F}^\star \phi]$, but there are no formulae of the form $\langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$ or $\langle\langle ? \rangle\rangle_{\leq y} \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$.

To see this, it suffices to realise that verifying any formula containing only one temporal operator (without $\langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$) is in NP, because there is a polynomial witness comprising the optimal coalition together with a winning strategy of all coalition players. Hence, for a formula containing multiple temporal operators, we can run the model checking algorithm described earlier, where we compute the set $\mathit{Sat}(\phi)$ using the NP oracle. For the case where the logic contains all operators, including $\langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$ and $\langle\langle ? \rangle\rangle_{\leq y} \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$, the problem is in NEXP $\cap$ coNEXP. In this case, we can reuse the idea of the proof of Theorem 1 (see Appendix C) by showing that there is an exponential size certificate. We give the proof in Appendix D.

4.5 Model Checking for rPATL*

Finally, in this section, we discuss model checking for the extended logic rPATL* introduced in Section 3.6. This can be done in a similar fashion to the logic PCTL* for Markov decision processes [8]. Model checking for an rPATL* formula ϕ can be performed as follows. Let $\phi_1, \phi_2, \cdots, \phi_n$ be a sequence of all (state) subformulae of ϕ, partially ordered by subsumption and where $\phi_n = \phi$. We compute $\mathit{Sat}(\phi_i)$ for each subformula ϕ_i in turn, starting from ϕ_1. If ψ_i is an rPATL formula, then we apply the rPATL model checking algorithm described above. Otherwise, it must be of the form $\langle\langle C \rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$, where ψ is an LTL formula.

For the latter case, we need to compute the optimal probabilities of satisfying LTL formula ψ for all states of the coalition game $\mathcal{G}_C$ and then compare these values with the bound q. Computing probabilities can be done in the following (standard) way. First, we translate ψ into a deterministic parity automaton with $\mathcal{O}(2^{2^{|\psi|}})$ states and $\mathcal{O}(2^{|\psi|})$ indices. Then, we build the product of the game $\mathcal{G}_C$ and the deterministic parity automaton, resulting in a stochastic two player zero-sum game with parity winning conditions. From the results of [15], we can compute the optimal value in $\mathcal{O}((|\mathcal{G}_C| \cdot 2^{2^{|\psi|}})^{2^{|\psi|}}) = \mathcal{O}(|\mathcal{G}_C|^{2^{|\psi|}} \cdot 2^{2^{2|\psi|}})$ time, which entails that model checking ψ can be done in 2EXPTIME. Hence, model checking rPATL* is 2EXPTIME-complete (where the lower bound follows from the fact that model checking LTL formulae for Markov decision processes is 2EXPTIME-hard [20]).

5 Implementation and Case Studies

Based on the techniques in this paper, we have built a probabilistic model checker for stochastic multi-player games, called PRISM-games[2] which is an extension of the PRISM tool [28]. For modelling of SMGs, we have adapted the PRISM modelling language. This allows multiple parallel components (called modules), which can either operate asynchronously or by synchronising over common action labels. Now, a model also includes a set of *players*, each of which controls transitions for a disjoint subset of the modules and/or action labels. Essentially, we retain the existing PRISM language semantics (for Markov decision processes), but, in every state, each nondeterministic choice belongs to one player. For the current work, we detect and disallow the possibility of concurrent decisions between players.

Our tool constructs an SMG from a model description and then executes the algorithms from Section 4 to check rPATL formulae. Currently, we have developed an explicit-state model checking implementation, which we show to be efficient and scalable for various large models. It would also be relatively straightforward to adapt PRISM's symbolic model checking engines for our purpose, if required.

5.1 Experimental Results

We have applied our tool to the analysis of four large case studies:

- *mdsm*: microgrid demand-side management;

[2] The tool is currently available from: `http://www.prismmodelchecker.org/games/`.

Case study [parameters]		SMG statistics			Model checking		
		Players	States	Transitions	Property type	Constr. (s)	Check (s)
mdsm $[N]$	5	5	743,904	2,145,120	$\langle\langle C\rangle\rangle \mathsf{R}^{r}_{\mathrm{max}=?}[\mathsf{F}^{\infty}\phi]$	14.5	61.9
	6	6	2,384,369	7,260,756		55.0	221.7
	7	7	6,241,312	19,678,246		210.7	1,054.8
cdmsn $[N]$	3	3	1,240	6,240	$\langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\,\mathsf{F}^{\leq k}\,\phi]$	0.2	0.2
	4	4	11,645	73,948		0.8	0.8
	5	5	100,032	760,430		3.2	6.4
investor $[vmax]$	10	2	10,868	34,264	$\langle\langle C\rangle\rangle \mathsf{R}^{r}_{\mathrm{max}=?}[\mathsf{F}^{c}\phi]$	1.4	0.7
	100	2	750,893	2,474,254		9.8	121.8
	200	2	2,931,643	9,688,354		45.9	820.8
team-form $[N]$	3	3	12,475	15,228	$\langle\langle C\rangle\rangle \mathsf{P}_{\mathrm{max}=?}[\mathsf{F}\,\phi]$	0.8	0.2
	4	4	96,665	116,464		1.6	0.9
	5	5	907,993	1,084,752		13.6	11.2

Table 1: Performance statistics for a representative set of models

- *cdmsn*: collective decision making for sensor networks;
- *investor*: the futures market investor example of [32];
- *team-form*: the team formation protocol of [17].

The first two of these case studies were developed solely for this work; the other two have been adapted from existing models.[3] Experimental results from the two new case studies are described in detail in the following sections. First, we show some statistics regarding the performance of our tool on a representative sample of models from the four case studies.

Table 1 shows model statistics (number of players, states and transitions) for three SMGs taken from each case study. It also gives the execution time for model checking a sample property on each one, divided into the time for model construction (building an SMG from its high-level modelling language description) and for executing the model checking algorithms of Section 4. Experiments were run on a 2.80GHz PC with 32GB RAM. Our aim here is not to provide a detailed analysis of the time required to model check different classes of rPATL formulae, but simply to give an indication of the scalability and efficiency of our algorithms and their implementation in PRISM-games.

We also briefly discuss how the performance of our model checking algorithms is affected by the speed of convergence of the underlying numerical computation methods. As mentioned in Section 4 above, our algorithms are mostly based on the evaluation of numerical fixpoints, termination of which is decided using a simple convergence test. More precisely, if X_s^k denotes the value computed for a state s in iteration k and ε represents a pre-specified *convergence threshold*, we terminate the computation when the maximum difference between values for successive iterations falls below ε, i.e. when:

$$\max_{s\in S} |X_s^k - X_s^{k-1}| < \varepsilon$$

Figure 2 illustrates, for several of the models from Table 1, how the total number of iterations of numerical computation required varies for different values of the convergence threshold ε. For the *cdmsn* example, we check a property of the form $\langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\mathsf{F}\,\phi]$ since the bounded property in Table 1 always requires exactly k iterations. We omit results for the *team-form* example since the corresponding

[3] Models and properties are at: `http://www.prismmodelchecker.org/files/fmsd-smg/`.

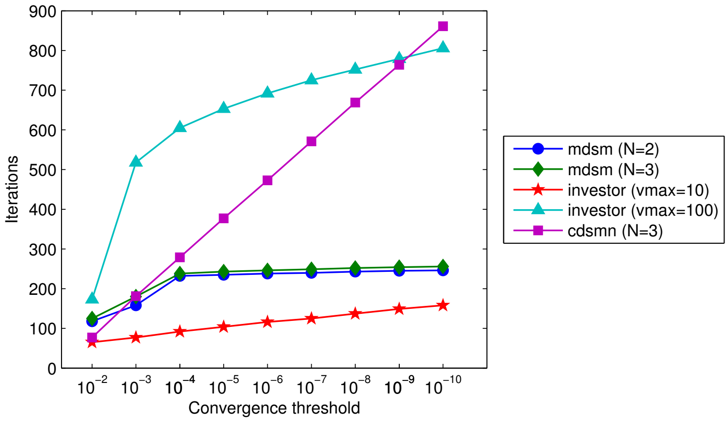


Fig. 2: Performance of numerical computation algorithms (number of iterations) for varying convergence thresholds ε.

SMGs do not contain cycles, meaning that varying ϵ has no effect. From the plots in Figure 2, a consistent pattern emerges: after (in some cases) initial irregularity for large values of ε, values of $\varepsilon = 10^{-n}$ for increasing values of n result in only a linear increase in the number of iterations required.

5.2 MDSM: Microgrid Demand-Side Management

Microgrid is an increasingly popular model for the future energy markets where neighbourhoods use electricity generation from local sources (e.g. wind/solar power) to satisfy local demand. The success of microgrids is highly dependent on *demand-side management*: active management of demand by users to avoid peaks. Thus, the infrastructure has to incentivise co-operation and discourage abuse in an environment which is highly decentralised and gives a high degree of control to users. Such systems are usually analysed using simulation studies, but these approaches can fail to uncover important features or weaknesses of the models. In this case study, we use rPATL model checking to analyse the MDSM infrastructure of [26] and identify an important incentive-related weakness.

The algorithm. The system in [26] consists of N *households* connected to a single *distribution manager* (DM). At every time-step, the DM randomly contacts a household for submission of a load for execution. Each load has an energy cost that is required for its execution. The probability of it generating a load is determined by a daily demand curve from [26] (see Figure 3). The duration of a load is random, between 1 and D time-steps. The cost of executing a load for a single step is the number of tasks currently running. Hence, the total cost increases quadratically with households executing more loads in a single step.

Each household follows a very simple algorithm, the essence of which is that, when it generates a load, if the cost is below an agreed limit c_{lim}, it executes it, and otherwise it only does so with a pre-agreed probability P_{start}. In [26], the *value* for each household in a time-step is measured by $V = \frac{\text{loads executing}}{\text{cost of execution}}$ and it is shown (through simulations) that, *provided every household sticks to this algorithm*,

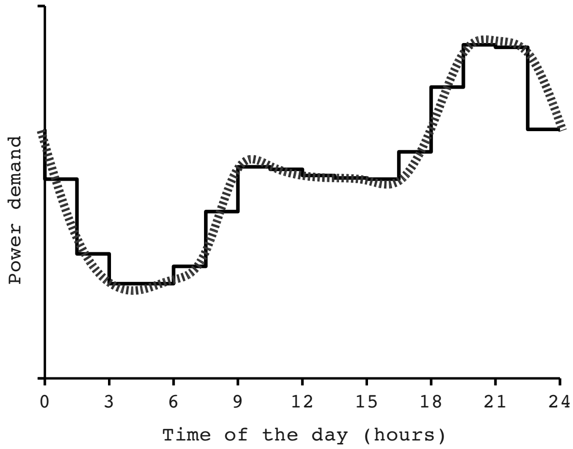


Fig. 3: MDSM energy demand curve from [26] and its piecewise approximation.

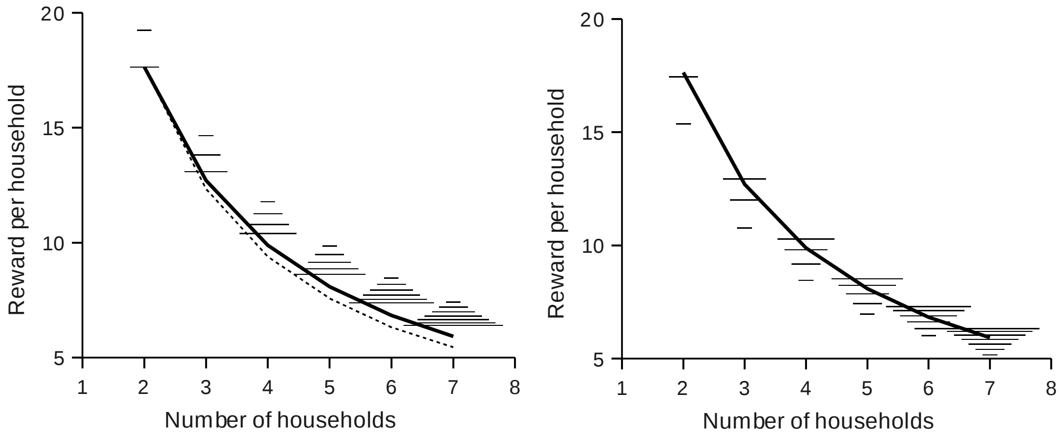
the peak demand and the total cost of energy are reduced significantly while still providing a good (expected) value V for each household.

Modelling and analysis. We modelled the system as an SMG with N players, one per household. We vary $N \in \{2, \ldots, 7\}$ (the size of the underlying SMG model is exponential in N and so we are unable to analyse systems with more than 7 households) and fix D=4 and c_{lim}=1.5. We analyse a period of 3 days, each consisting of 16 time-steps (using a piecewise approximation of the daily demand curve, shown in Figure 3). First, as a benchmark, we assume that all households follow the algorithm of [26]. We define a reward structure r_i for the value V for household i at each step, and let $r_C = \sum_{i \in C} r_i$ be the total reward for a set of households C. To compute the expected value per household, we use the rPATL query:

$$\tfrac{1}{|C|} \langle\langle C \rangle\rangle \mathsf{R}^{r_C}_{\max=?} [\, \mathsf{F}^0 \text{ time=max\_time} \,]$$

Initially, we fix C to be the set Π of all N players (households). We use this to determine the optimal value of P_{start} achievable by a memoryless strategy for each player, which we will then fix. These results are shown by the bold lines in Figure 4. We also plot (as a dotted line) the values obtained if no demand-side management is applied.

Next, we consider the situation where the set of households C is allowed to deviate from the pre-agreed strategy, by choosing to ignore the limit c_{lim} if they wish. We check the same rPATL query as above, but now varying C to be coalitions of different sizes, $C \in \{\{1\}, \{1, 2\}, \ldots, \Pi\}$. The resulting values are also plotted in Figure 4a, shown as horizontal dashes of width proportional to $|C|$: the shortest dash represents individual deviation, the longest is a collaboration of all households. The former shows the maximum value that can be achieved by following the optimal collaborative strategy, and in itself presents a benchmark for the performance of the original algorithm. The key result is that deviations by individuals or small coalitions guarantee a better expected value for the households than *any* larger collaboration: a highly undesired weakness for an MDSM system.



(a) Original version.

(b) Version with punishment.

Fig. 4: Expected value per household for MDSM. The bold line shows all households following the algorithm of [26]; the dotted line shows the case without DSM. Horizontal dashes show deviations by collaborations of increasing size (shortest dash: individual deviation; longest dash: deviation of all households).

Fixing the algorithm. We propose a simple punishment mechanism that addresses the problem: we allow the DM to cancel *one* job per step if the cost exceeds c_{lim}. The intuition is that, if a household is constantly abusing the system, its job could be cancelled. Results for the same set of rPATL queries on a revised model that incorporates the punishment mechanism are shown in Figure 4b. We see that the modification of the algorithm inverts the incentives. The best option now is full collaboration and small coalitions who deviate *cannot* guarantee better expected values any more.

5.3 CDMSN: Collective Decision Making for Sensor Networks

Sensor networks comprise a set of low-power, autonomous devices which must act collaboratively in order to achieve a particular goal. Strategic analysis of such systems can help to establish performance boundaries (e.g., find the best value achievable by the sensor network assuming full collaboration) and analyse performability (e.g., look at the performance of the system in the presence of failure or unexpected behaviour of some nodes). In this case study, we illustrate the use of rPATL model checking to aid the analysis and design of such systems by studying a distributed consensus algorithm for sensor networks [33].

The algorithm. There are N sensors deployed in an environment having a set of targets $K = \{k_1, k_2, \dots\}$, each with *quality* $Q_{k_j} \in [0, 1]$. The goal is for the sensors to agree on a target with maximum Q_{k_j}. Each sensor i stores a *preferred* target $p_i \in K$, its quality Q_{p_i} and an integer $l_i \in \{1, \dots, L\}$ to represent *confidence* in the preference. The algorithm has the following non-linearity parameters:

- η, which affects the relative perceived quality of targets when a sensor is comparing one to the other;

- λ, which affects the relative perceived quality of targets when two sensors are comparing their preferred targets;
- γ, which affects the relative weight of the confidence level when two sensors are comparing their preferred targets.

A sensor has three actions: *sleep*, *explore* and *communicate*. As proposed by [33], each sensor repeatedly *sleeps* for a random time t and then either *explores* (with probability P_{exp}) or *communicates*. For the *explore* action, sensor i picks a target $k \in K$ uniformly at random and with probability $P_k = Q_k^\eta/(Q_k^\eta + Q_{p_i}^\eta)$ switches its preference (p_i) to k and resets confidence to 1. To *communicate*, it compares its preference with that of a random sensor j. If they agree, both confidences are increased. If not, with probability

$$P_{\mathrm{s}} = \frac{Q_{p_j}^\lambda l_j^\gamma}{Q_{p_j}^\lambda l_j^\gamma + Q_{p_i}^\lambda l_i^\gamma},$$

sensor i switches preference to p_j, resets confidence to 1 and increases sensor j's confidence; with probability $1{-}P_{\mathrm{s}}$, the roles of sensors i and j are swapped, i.e. sensor j switches preference to p_i, resets confidence to 1 and increases sensor i's confidence.

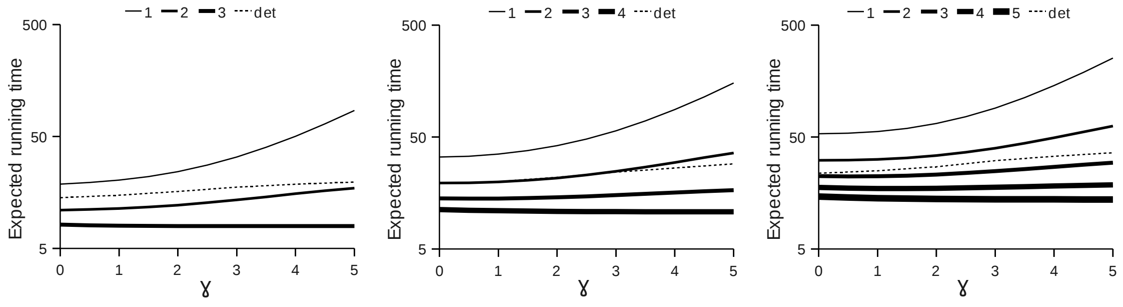
The purpose of the system is to locate and agree upon the target having the best quality by striking a balance between exploration of the environment (i.e., sensing) and communication with other sensors. Intuitively, a 'good' strategy for sensors should be to actively communicate when they believe they have found the best quality target and to explore otherwise. An example of an antagonistic strategy could be one which actively communicates to advertise a low quality target, thus not only polluting the system with false information, but also draining the resources of the sensor network.

Modelling and analysis. We have modelled the system as an SMG with N players, one per sensor. We consider models with $N{=}3, 4, 5$, three targets $K{=}\{k_1, k_2, k_3\}$ with qualities $Q_{k_1}{=}1$, $Q_{k_2}{=}0.5$, $Q_{k_3}{=}0.25$ and two confidence levels $l_i \in \{1, 2\}$. As in [33], we assume a random scheduling and fix parameters $\eta{=}1$ and $\lambda{=}1$. In [33], two key properties of the algorithm are studied: *speed of convergence* and *robustness*. We use our rPATL framework to evaluate both of these and explore alternative strategies for sensors (i.e. allowing sensors to execute any action when active). We also assume that only a subset C of the sensors are under our control, e.g. because the others are faulty. We use rPATL queries (with coalition C) to optimise performance, under the worst-case assumption about the other sensors.

First, we study the *speed of convergence* and the influence of parameter γ upon it. In [33], it is shown that increasing γ improves the speed of convergence to *a decision* and stability of it. Figure 5 shows the expected running time to reach the *best decision* (i.e. select k_1) for various values of γ and sizes of the coalition C. We use the reward structure: $r(s) = 1$ for all $s \in S$ and rPATL query:

$$\langle\langle C \rangle\rangle \mathsf{R}^r_{\min=?}[\,\mathsf{F}^\infty \textstyle\bigwedge_{i=1}^{|\Pi|} p_i = k_1\,].$$

where $C \in \{\{1\}, \{1,2\}, \ldots, \Pi\}$. Figure 5 also shows the performance of the original algorithm [33] (line 'det'). We make several important observations. First, if we lose control of a few sensors (e.g. because a fault occurs), we can still guarantee a

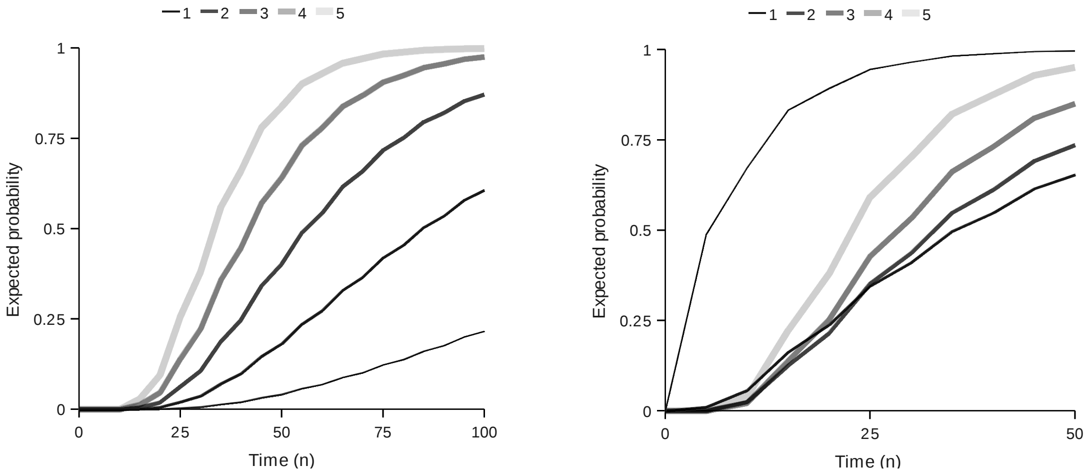


(a) $N = 3$

(b) $N = 4$

(c) $N = 5$

Fig. 5: Expected running time until the selection of the best quality target for different models and increasing sizes of coalition C. Dotted lines show optimal performance that can be achieved using the original algorithm from [33].



(a) Probability to select k_1 within 10 steps is greater than 0.9.

(b) Expected energy usage for coalition to select k_1 is less than $0.5mJ$.

Fig. 6: Minimum probability to *recover* from a state where all sensors prefer the lowest quality target, k_3, within n steps for different coalition sizes. Graphs (a) and (b) show results for two types of *recovery state* (see captions), with $\gamma = 2$.

good convergence time, indicating the fault tolerance potential of the system. On the other hand, the original version performs almost as well as the optimal case for large coalitions.

Secondly, we consider *robustness*: the ability of the coalition C to recover from a ‘bad’ state (i.e. $\bigwedge_{i=1}^{|\Pi|} p_i = k_3$) to a ‘good’ state in n steps; this can be specified by rPATL formula $\langle\langle C \rangle\rangle \mathsf{P}_{\max=?}[\,\mathsf{F}^{\leq n}\, \phi_{good}]$, where ϕ_{good} represents ‘good’ states. Below, we give two interpretations of a ‘good’ state and show that the results for them are quite different. Note that the formulae below characterise a strategy for coalition C that consists of two parts: first, the strategy from the outer $\langle\langle C \rangle\rangle$ recovers to a ‘good’ state; and second, once the ‘good’ state is reached, the strategy of the coalition switches to the one defined by the inner $\langle\langle C \rangle\rangle$ operator.

(1) For a 'good' state, there exists a strategy for coalition C to make all sensors, *with probability* > 0.9*, select* k_1 *within 10 steps.* So robustness in rPATL is:

$$\langle\langle C\rangle\rangle \mathsf{P}_{\max=?}[\,\mathsf{F}^{\leq n}\ \langle\langle C\rangle\rangle \mathsf{P}_{>0.9}[\,\mathsf{F}^{\leq 10}\ \textstyle\bigwedge_{i=1}^{|\Pi|} p_i = k_1]]\,.$$

(2) For a 'good' state, there exists a strategy for coalition C to make all sensors *select* k_1 *while using less than* $0.5mJ$ *of energy.* We use a reward structure r_C representing energy usage by sensors in C: power consumption is $10mW$ for each communication and $1mW$ for each exploration, and each activity takes 0.1s. Then, robustness in rPATL is:

$$\langle\langle C\rangle\rangle \mathsf{P}_{\max=?}[\,\mathsf{F}^{\leq n}\ \langle\langle C\rangle\rangle \mathsf{R}^{r_C}_{<50}[\mathsf{F}^{c}\ \textstyle\bigwedge_{i=1}^{|\Pi|} p_i = k_1]]\,.$$

Figure 6 shows, for each definition and for a range of values of n, the *worst-case* (minimum) value for the rPATL query from all possible 'bad states'. For (1), the results are intuitive: the larger the coalition, the faster it recovers. For (2), however, the one-sensor coalition outperforms all others. Also, we see that, in the early stages of recovery, 2-sensor coalitions outperform larger ones. This shows that small coalitions can be more resource efficient in achieving certain goals.

Note that the performance results detailed here are in a complete-information setting, which implicitly assumes that the members of the coalition (i.e., non-faulty sensors) have knowledge of which sensors are faulty and can adjust their behaviour accordingly, and therefore the values provided by our analysis are upper bounds on the performance that can be achieved.

6 Conclusions

We have designed and implemented a framework for automatic verification of systems with both probabilistic and competitive behaviour, based on stochastic multi-player games. We proposed a new temporal logic rPATL, designed model checking algorithms, implemented them in a tool and then used our techniques to identify unexpected behaviour in several large case studies.

There are many interesting directions for future work in this area. Firstly, we plan to further develop our probabilistic model checker PRISM-games, including synthesis of strategies for rPATL and analysis of wider classes of properties for SMGs (e.g., reward operators dealing with limit averages and discounted sums). Secondly, we would like to investigate extensions of our techniques to incorporate partial-information strategies or more complex solution concepts such as Nash, subgame-perfect or secure equilibria. We would also like to explore the applicability of our work as an underlying solution framework for more complex analysis of multi-agent systems.

Acknowledgements

The authors are partially supported by ERC Advanced Grant VERIWARE, the Institute for the Future of Computing at the Oxford Martin School and EPSRC grant EP/F001096/1. Vojtěch Forejt is supported by a Royal Society Newton Fellowship. We also thank the anonymous referees for various helpful comments.

Appendix: Proofs

This appendix contains proofs for the results stated in the text. We begin by stating some known results that we will require later.

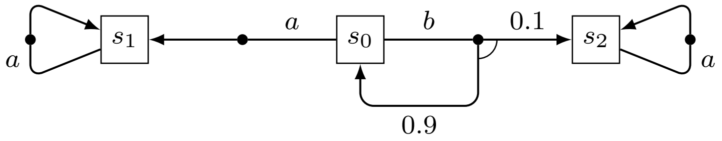
Theorem 2 ([6,12]) *The following statements hold:*

1. *Memoryless deterministic strategies suffice for achieving minimum/maximum values in a state for extended reachability, Büchi, and coBüchi objectives in stochastic two-player zero-sum games.*
2. *Finding minimum/maximum values in a state for Markov decision processes (MDPs) for extended reachability, Büchi, and coBüchi objectives can be done in polynomial time.*

A Proofs for Optimality of Expected Rewards

A.1 Finite-memory Strategies for $\star = 0$

We first show that finite-memory strategies are *required* for optimality of expected rewards of type $\star = 0$, i.e. for optimal values of $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r, 0, T)]$. Later, in the proof of Lemma 3 we show that the finite memory is indeed sufficient. Let us consider the following example:



The target set is $T = \{s_1\}$ and the reward structure r assigns 1 to s_0 and 0 to the other states. We analyse the optimal value of $\mathit{rew}(r, 0, T)$ in s_0. Let σ be a memoryless strategy that in s_0 picks a with probability x and b with probability $1 - x$. The reward obtained is then:

$$\begin{aligned}
&\sum_{i=1}^{\infty} i \cdot 0.9^{i-1} \cdot (1-x)^{i-1} \cdot x = \\
&= \frac{x}{(0.9 - 0.9 \cdot x)} \cdot \sum_{i=1}^{\infty} i \cdot (0.9 - 0.9 \cdot x)^i \\
&= \frac{x}{(0.9 - 0.9 \cdot x)} \cdot \frac{(0.9 - 0.9 \cdot x)}{\big(1 - (0.9 - 0.9 \cdot x)\big)^2} \\
&= \frac{x}{(0.1 + 0.9 \cdot x)^2}
\end{aligned}$$

which, for any x, is lower than $\frac{25}{9}$.

Now consider the strategy σ' that is deterministic, and picks b on the first 8 visits to s_0 and then a on the 9th visit. The value under this strategy is:

$$9 \cdot 0.9^8 \approx 3.8 > \frac{25}{9}$$

Remark An optimal (memoryless deterministic) strategy in s_0 for both $\star = \infty$ and $\star = c$ is to take the action b and thus achieve values ∞ and 10, respectively.

A.2 Memoryless Strategies for $\star = \{\infty, c\}$

Secondly, we prove that memoryless (deterministic) strategies suffice for optimality of the expected reward $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)]$ for types $\star = \{\infty, c\}$.

If the expected value is infinite, then memoryless deterministic strategies suffice by Theorem 2 because this cases reduces to the problem of reaching a state where the expected value is infinite with positive probability. The states $s \in T$ get value 0 by definition. Otherwise, the values $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)]$ satisfy:

$$\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)] = r(s) + \mathrm{opt}^s_{a\in A(s)} \sum_{s'\in S} \varDelta(s,a)(s') \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s'}[\mathit{rew}(r,\star,T)] \qquad (3)$$

Let $A_{\mathrm{opt}}(s)$ be the set of actions that realise the optimum in s, where opt is max or min, for players 1 and 2, respectively; similarly opt^s is max if $s \in S_1$ and min if $s \in S_2$.

We first analyse the case $\star = \infty$. Any strategy $\sigma_1^\infty \in \varSigma_1$ that in s picks the action from $A_{\mathrm{opt}}(s)$ is optimal. For player 2, any strategy $\sigma_2^\infty \in \varSigma_2$ is optimal if it picks the action from $A_{\mathrm{opt}}(s)$ in s such that T is reached almost surely under any counter-strategy for player 1.

Next, assume $\star = c$ and let $T_0 = \{s \mid \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,c,T)] = 0\}$. To optimise $\mathit{rew}(r,c,T)$, we fix $\sigma_1^c \in \varSigma_1$ that uses an action $A_{\mathrm{opt}}(s)$ in s and ensures that T_0 is reached almost surely. For player 2, any strategy $\sigma_2^c \in \varSigma_2$ is optimal if it picks an action from $A_{\mathrm{opt}}(s)$ in s.

Proof of correctness of definitions of strategies. Given a state s and a strategy σ_1 for player 1, we denote:

$$\mathit{err}^{\sigma_1}(s) = \frac{\min_{\sigma_2\in\varSigma_2} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)]}{\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)]}$$

where we assume $\mathit{err}^{\sigma_1}(s) = 1$ if the denominator is 0. Observe that we have $\mathit{err}^{\sigma_1}(s) \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)] = \min_{\sigma_2\in\varSigma_2} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,\star,T)]$.

Let $\star = c$. We prove that the maximiser's strategy $\sigma = \sigma_1^c$ defined above is optimal. Assume, for a contradiction, that it is not, i.e. $\mathit{err}^\sigma(s) < 1$ for some s. For all s, we have:

$$\mathit{err}^\sigma(s) \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,c,T)] = r(s) + \sum_{s'\in S} \varDelta(s,\sigma(s))(s') \cdot \mathit{err}^\sigma(s') \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s'}[\mathit{rew}(r,c,T)] \quad (4)$$

and, for all $s \in S_2$, there must be an action a such that:

$$\mathit{err}^\sigma(s) \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,c,T)] = r(s) + \sum_{s'\in S} \varDelta(s,a)(s') \cdot \mathit{err}^\sigma(s') \cdot \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s'}[\mathit{rew}(r,c,T)] \quad (5)$$

Fix s such that $\mathit{err}^\sigma(s) < 1$ is minimal. Thanks to equations (3), (4) and (5), we get that the value must also be minimal for all successors of s. However, this implies that T_0 is not reached with probability equal to 1 because, in every $s' \in T_0$, we have $\mathit{err}^\sigma(s') = 1$.

The other cases (σ_2^c, σ_1^∞ and σ_1^∞) can be proved analogously.

B Proofs of Correctness for Section 4.3

In this section, we prove the correctness of the methods given in Section 4.3 for computing $\mathit{rew}(r,\star,T)$ for the cases $\star = \{c, \infty, 0\}$.

B.1 Proof of Correctness for $\star = c$

Let us first consider the states with infinite value. Recall that we denote by $\mathit{inf}(a_{\mathrm{rew}})$ the set of paths that visit a state with positive reward infinitely often (and thus get infinite reward). If, for a state s, there is $\sigma_1 \in \varSigma_1$ such that the probability $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}(a_{rew}))$ is positive for all $\sigma_2 \in \varSigma_2$, then the strategy σ_1 itself yields the infinite reward. In the other direction, suppose that for every $\sigma_1 \in \Sigma_1$ there is some $\sigma_2 \in \Sigma_2$ such that $\mathrm{Pr}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}(a_{rew}))$ is equal to zero. It is straightforward to extend the results of [22] and prove that, for every σ_1, a strategy σ_2 exists which in addition ensures that the expected number of visits to a state satisfying a_{rew} is finite and bounded from above. The rest follows easily because the rewards assigned to states are also bounded from above.

Let us now consider finite values. Because of the assumption that no reward is accumulated after visiting a target state, we can change the random variable and use $\sum_{j\in\mathbb{N}} r(st_\lambda(j))$ instead of $\mathit{rew}(r,c,T)$. It can be shown by induction that the expected value w.r.t. this variable can be obtained as $\lim_{i\to\infty} f_s(i)$ where:

$$f_s(i) = \begin{cases} 0 & \text{if } i = 0 \\ r(s) + \mathrm{opt}^s_{a\in Act(s)} \sum_{s'\in a_S} \Delta(s,a)(s') \cdot f_{s'}(i-1) & \text{otherwise} \end{cases} \tag{6}$$

We can then apply the Kleene fixpoint theorem and prove that $\lim_{i\to\infty} f_s(i)$ is equal to the least fixpoint of the equations (2).

B.2 Proof of Correctness for $\star = \infty$

First, observe that if a state s is assigned infinite value in the initial step, then we indeed have $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)] = \infty$ by definition. We prove the correctness for the other values. Let $u : S \to \mathbb{Q}$ be a function that assigns to each s a value such that $u(s) \geq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$. Recall that we compute values of equations (2) by value iteration, i.e. we compute:

$$f(s)(i) = \begin{cases} 0 & \text{if } s \in T \\ u(s) & \text{if } i = 0 \\ r(s) + \mathrm{opt}^s_{a\in A(s)} \sum_{s'\in S} \Delta(s,a)(s') \cdot f(s')(i-1) & \text{otherwise} \end{cases}$$

for sufficiently large i, and we show that $\lim_{i\to\infty} f(s)(i) = \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$.

Let us consider auxiliary functions rew^i_u which assign numbers to paths as follows:

$$\mathit{rew}^i_u(\lambda) = \begin{cases} \sum\limits_{j<k} r(st_\lambda(j)) & \exists k \leq i : st_\lambda(k) \in T \wedge \forall j < k : st_\lambda(j) \notin T, \\ \sum\limits_{j<i} r(st_\lambda(j)) + u(st_\lambda(i)) & \text{otherwise.} \end{cases}$$

Intuitively, the function rew^i_u alters the definition of $\mathit{rew}(r,\infty,T)$ by assigning rewards given by r for the first i steps, and then assigning the reward given by u, if the target has not been reached yet. One can easily prove by induction that the value of $f(s)(i)$ is equal to $\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}^i_u]$.

We need to show that $\lim_{i\to\infty} f(s)(i) \geq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$. This can be done by inductively showing that $f(s)(i) \geq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$ for every i. The base case $i = 0$ follows from the definition of f and u, and the inductive steps follow by monotonicity of the function f.

Furthermore, we show that $\lim_{i\to\infty} f(s)(i) \leq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$. Let $\sigma_{\min} \in \Sigma_2$ be a memoryless strategy satisfying $\max_{\sigma\in\Sigma_1} \mathbb{E}^{\sigma,\sigma_{\min}}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)] = \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$, i.e. $\sigma_{\min}$ is the optimal minimising strategy for player 2. Let $\tau(i) = \min_{\sigma\in\Sigma_1} \mathrm{Pr}^{\sigma,\sigma_{\min}}_s(\{\lambda \in \Omega_{\mathcal{G}_C,s} \mid \exists j \leq i : st_\lambda(j) \in T\})$ be the minimal probability with which we end in T within i steps when playing according to $\sigma_{\min}$. We have $\lim_{i\to\infty} \tau(i) = 1$, because otherwise player 1 would have a strategy to prevent the target from being reached almost surely and the reward obtained would be infinite. Thus, we have $\max_{\sigma\in\Sigma_1} \mathbb{E}^{\sigma,\sigma_{\min}}_s[\mathit{rew}^i_u] \leq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)] + (1-\tau(i)) \cdot K$ where $K = \max_{s\in S} u(s)$. As we let i go to ∞, the second summand diminishes, and so $f(s)(i) = \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}^i_u] \leq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,\infty,T)]$. □

B.3 Proof of Correctness for $\star = 0$

Lemma 1 $\sup_{\sigma_1\in\Sigma_1}\inf_{\sigma_2\in\Sigma_2}\mathbb{E}_s^{\sigma_1,\sigma_2}[\mathit{rew}(r,0,T)] = \infty$ *iff there is* $\sigma_1\in\Sigma_1$ *such that for all* $\sigma_2\in\Sigma_2$ $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}^t(a_{\mathit{rew}})) > 0$.

Proof In the direction $\Leftarrow$, let $q \in \mathbb{R}$ be any number. Player 1's strategy σ to ensure that the expected reward achieved is at least q works as follows. Suppose σ_1 ensures $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}^t(a_{\mathit{rew}})) > p$ for all σ_2. By [22], we can safely assume that $p > 0$. The strategy σ mimics a strategy $\sigma_1 \in \Sigma_1$ if the history λ satisfies $\mathrm{r}(\lambda) < \frac{q}{p\cdot x^{|S|}}$ where x is the minimal probability that occurs in the game. When $\mathrm{r}(\lambda)$ exceeds this bound and the formula $\mathsf{P}_{>0}[\mathsf{F}\,t]$ is satisfied in the last state of λ, the strategy σ changes its behaviour and maximises the probability to reach T. Because memoryless deterministic strategies are sufficient for both players for reachability queries, σ can ensure that T is reached with probability at least $x^{|S|}$ from λ. The rest is a simple computation.

Let us analyse the direction $\Rightarrow$. Similarly to the $\star = c$ case, we can show that, if for every $\sigma_1 \in \Sigma_1$ there is $\sigma_2 \in \Sigma_2$ such that $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathit{inf}(a^t_{\mathit{rew}}))$ is equal to zero, then there is σ_2 which ensures that the expected number of visits to a state satisfying a^t_{rew} is finite. The rest follows as in $\star = c$; we only need to further consider that if the state satisfies a_{rew} but not $\mathsf{P}_{>0}[\mathsf{F}\,t]$ (i.e. it gets nonzero reward but is not labelled with a^t_{rew}), then the reward achievable by player 1 in such a state is 0.

Given the state s, we denote the set of actions which can be taken by the strategy which achieved maximum probability to reach T by $A(s,T)$. We first show that, if player 1 wants to maximise the expected reward w.r.t. $\mathit{rew}(r,0,T)$ using only actions from $A(s,T)$ in each state, he can do so using a memoryless deterministic strategy.

Lemma 2 *Let* $\Sigma_1^T \subseteq \Sigma_1$ *contain all strategies that use only the actions from* $A(s,T)$ *and* $\forall\sigma_1 \in \Sigma_1^T\colon \min_{\sigma_2\in\Sigma_2}\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathsf{F}\,t) = \Pr^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\,t)$. *There is a memoryless deterministic strategy* $\sigma_1^* \in \Sigma_1^T$ *satisfying:*

$$\min_{\sigma_2\in\Sigma_2}\mathbb{E}^{\sigma_1^*,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T)] = \max_{\sigma_1\in\Sigma_1^T}\min_{\sigma_2\in\Sigma_2}\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T)]\,.$$

Proof Assume the game is restricted so that the only actions available in s are $A(s,T)$ for all s. We first create a new reward structure r' defined by $r'(s) = r(s)\cdot\Pr^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\,t)$. We show that, for all $\sigma_1 \in \Sigma_1^T$ and $\sigma_2 \in \Sigma_2$ with $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathsf{F}\,t) = \Pr^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\,t)$, we have that:

$$\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r',c,T)] = \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T)]\,,$$

from which the lemma follows directly, as memoryless deterministic strategies suffice for achieving the optimal value of $\mathit{rew}(r',c,T)$ (see the proof in Appendix A.2).

Let $\Omega_{\mathcal{G}_C,s}(T) \overset{\text{def}}{=} \{\lambda \in \Omega_{\mathcal{G}_C,s} \mid \exists i : st_\lambda(i) \in T\}$, and $t(\lambda) = \min_{i\in\mathbb{N}} st_\lambda(i) \in T$. For any strategy profile σ_1, σ_2 such that $\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\mathsf{F}\, t) = \Pr^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t)$,

$$\begin{aligned}
\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,0,T)] &= \int_{\Omega_{\mathcal{G}_C,s}} rew(r,0,T)(\lambda) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \int_{\Omega_{\mathcal{G}_C,s}(T)} \sum_{n=0}^{t(\lambda)} r(st_\lambda(n)) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \int_{\Omega_{\mathcal{G}_C,s}(T)} \sum_{n=0}^{\infty} r(st_\lambda(n)) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \sum_{n=0}^{\infty} \int_{\Omega_{\mathcal{G}_C,s}(T)} r(st_\lambda(n)) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \sum_{n=0}^{\infty} \sum_{s'\in S} r(s') \cdot \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(st_\lambda(n)=s' \wedge \lambda \models \mathsf{F}\, t) \\
&= \sum_{n=0}^{\infty} \sum_{s'\in S} r(s') \cdot \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(st_\lambda(n)=s') \cdot \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\lambda \models \mathsf{F}\, t \mid st_\lambda(n)=s') \\
&= \sum_{n=0}^{\infty} \sum_{s'\in S} r(s') \cdot \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(st_\lambda(n)=s') \cdot \Pr^{\max,\min}_{\mathcal{G}_C,s'}(\mathsf{F}\, t) \\
&= \sum_{n=0}^{\infty} \sum_{s'\in S} r'(s') \cdot \Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(st_\lambda(n)=s') \\
&= \sum_{n=0}^{\infty} \int_{\Omega_{\mathcal{G}_C,s}} r'(st_\lambda(n)) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \int_{\Omega_{\mathcal{G}_C,s}} rew(r',c,T)(\lambda) d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s} \\
&= \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r',c,T)].
\end{aligned}$$

This completes the proof. □

Below, given a path h, we use $\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,0,T) \mid h]$ to denote the *conditional* expectation of $rew(r,0,T)$ on infinite paths initiated in h, i.e.:

$$\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,0,T) \mid h] = \frac{\int_{\{\lambda \mid \lambda \text{ starts with } h\}} \mathrm{r}(\lambda)\, d\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}}{\Pr^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}(\{\lambda \mid \lambda \text{ starts with } h\}}$$

Lemma 3 *For each state $s \in S$, there exists a finite-memory strategy σ^* for player 1 which maximises the expected reward $rew(r,0,T)$ from the state s. In particular, there exists some bound B such that for $r(h) \geq B$, $\sigma^*(h)$ becomes memoryless.*

Proof Fix two strategies $\sigma_1 \in \Sigma_1$ and $\sigma_2 \in \Sigma_2$. For each state $s \in S$ and a path $h = s_0a_0s_1\ldots s_n$ ending in s' we have that:

$$\begin{aligned}
\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[rew(r,0,T) \mid h] &= \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s}[rew(r,0,T) + r(h)] \\
&= \int_{\{\lambda\in\Omega_{\mathcal{G}_C,s} \mid \lambda\models \mathsf{F}\, t\}} r(h) d\Pr^{\sigma_1,\sigma_2}_{s} + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[rew(r,0,T)] \\
&= \Pr^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) \cdot r(h) + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[rew(r,0,T)]
\end{aligned}$$

where $rew(r,0,T) + r(h)$ is a random variable assigning $rew(r,0,T)(\lambda) + r(h)$ to a path h reaching T, and 0 otherwise; and where $\sigma_i^h(h') = \sigma_i(s_0a_0s_1\ldots s_{n-1}a_{n-1}\cdot h')$.

Given a state s, we use $PR_s^{\underline{\max}}$ to denote the maximal reachability probability to reach T under the strategies for which at s, actions in $A(s,T)$ are disallowed for a single step, i.e.:

$$PR_s^{\underline{\max}} = \max_{a \in A(s) \setminus A(s,T)} \sum_{s' \in S} \Delta(s,a)(s') \cdot \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s'}(\mathsf{F}\, t)$$

Intuitively, $PR_s^{\underline{\max}}$ denotes the "second" maximal reachability probability. Below, we assume that $A(s,T) \neq \Delta(s)$. Define:

$$B_s = \frac{\mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,c,T)]}{\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s}(\mathsf{F}\, t) - PR_s^{\underline{\max}}}$$

Let $B = \max_{s \in S} B_s$. We show that, on paths h ending in s and satisfying $\mathrm{r}(h) > B$, no optimal strategy of player 1 can use actions from $A(s) \setminus A(s,T)$ and, together with Lemma 2, we obtain the statement of this lemma.

Let h be a path ending in $s_n \in S_1$ and satisfying $\mathrm{r}(h) > B$. Assume $\sigma_1(h)$ deterministically chooses action from $A(s) \setminus A(s,T)$ (for randomised choices the argument follows analogously). By above we have, for any $\sigma_2 \in \Sigma_2$:

$$\begin{aligned}
&\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T) \mid h] \\
&= \mathrm{Pr}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s}(\mathsf{F}\, t) \cdot \mathrm{r}(h) + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,0,T)] \\
&\leq PR_s^{\underline{\max}} \cdot \mathrm{r}(h) + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,0,T)] \\
&= \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) \cdot \mathrm{r}(h) - (\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) - PR_{s_n}^{\underline{\max}}) \cdot \mathrm{r}(h) + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,0,T)] \\
&< \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) \cdot \mathrm{r}(h) - (\mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) - PR_{s_n}^{\underline{\max}}) \cdot B + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,0,T)] \\
&\leq \mathrm{Pr}^{\max,\min}_{\mathcal{G}_C,s_n}(\mathsf{F}\, t) \cdot \mathrm{r}(h) - \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,c,T)] + \mathbb{E}^{\sigma_1^h,\sigma_2^h}_{\mathcal{G}_C,s_n}[\mathit{rew}(r,0,T)] \\
&\leq \mathbb{E}^{\max,\min}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T) \mid h]
\end{aligned}$$

which contradicts that σ_1 is optimal.

Clearly, the strategy optimising $\mathit{rew}(r,0,T)$ is of finite-memory with upper bound B on the memory needed. □

By the equalities from the proof of Lemma 2 and by Lemma 3, the procedure described in step 2 of the algorithm on page 13 is correct. The procedure from step 3 of the algorithm is correct because, for all paths h, we have that:

$$\mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T) \mid h] = \max_{a \in A(s)} \sum_{s' \in S} \Delta(s,a)(s') \cdot \mathbb{E}^{\sigma_1,\sigma_2}_{\mathcal{G}_C,s}[\mathit{rew}(r,0,T) \mid h{\cdot}a{\cdot}s'].$$

C Proof of Theorem 1

Theorem 1(a). Let φ be a rPATL formula with no $\langle\langle C \rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$ operator and where k for the temporal operator $\mathsf{U}^{\leq k}$ is given in unary. The problem of deciding whether the formula is satisfied in s is in NP ∩ coNP.

Proof By equivalences such as the one in equation (1) on page 8, we can assume that all probabilistic and reward operators only contain bounds $\geq$ or $>$, so in the proof we assume $\bowtie \in \{>, \geq\}$.

Let $\varphi_1, \varphi_2, \ldots, \varphi_n$ be the sequence of all state formulae occurring in φ. Also, if φ_i's outermost operator is temporal, let C_i denote the outermost coalition in φ_i, and Σ_j^C denote the set of all *memoryless deterministic* strategies for player j in the coalition game $\mathcal{G}_C$.

We show that the problem is in NP ∩ coNP by describing a polynomial-size *certificate* c that allows us to check that a formula is (not) satisfied. The certificate c is a function that assigns an element of $\Sigma_1^{C_i} \cup \Sigma_2^{C_i}$ to each tuple (i, s) where $s \in S$ and φ_i is a formula whose outermost operator is temporal:

- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$ and $s \models \varphi_i$, then: $c(i, s) = \sigma_1$ for $\sigma_1 \in \Sigma_1^C$ such that $\min_{\sigma_2 \in \Sigma_2} \Pr_{\mathcal{G}_C,s}^{\sigma_1,\sigma_2}(\psi) \bowtie q$ holds.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$ and $s \not\models \varphi_i$, then: $c(i, s) = \sigma_2$ for $\sigma_2 \in \Sigma_2^C$ such that $\max_{\sigma_1 \in \Sigma_1} \Pr_{\mathcal{G}_C,s}^{\sigma_1,\sigma_2}(\psi) \bowtie q$ does *not* hold.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^\star \phi]$ and $s \models \varphi_i$, then: $c(i, s) = \sigma_1$ for $\sigma_1 \in \Sigma_1^C$ such that $\min_{\sigma_2 \in \Sigma_2} \mathbb{E}_{\mathcal{G}_C,s}^{\sigma_1,\sigma_2}[\mathit{rew}(r, \star, \mathit{Sat}(\phi))] \bowtie x$ holds.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^\star \phi]$ and $s \not\models \varphi_i$, then: $c(i, s) = \sigma_2$ for $\sigma_2 \in \Sigma_2^C$ such that $\max_{\sigma_1 \in \Sigma_1} \mathbb{E}_{\mathcal{G}_C,s}^{\sigma_1,\sigma_2}[\mathit{rew}(r, \star, \mathit{Sat}(\phi))] \bowtie x$ does *not* hold.

The existence of the strategies assigned by c follows from Theorem 2 and from Appendix A.2.

To check the certificate in polynomial time, we compute $\mathit{Sat}(\varphi')$ for all state subformulae φ' of φ, traversing the parse tree of φ bottom-up. Suppose that we are analysing a formula φ' and that we have computed $\mathit{Sat}(\varphi'')$ for all state subformulae φ'' of φ'. If φ' is an atomic proposition or its outermost operator is a boolean connective, we construct $\mathit{Sat}(\varphi'')$ in the obvious way. Otherwise:

$$\mathit{Sat}(\varphi') = \{s \mid c(i, s) \text{ is a strategy for the first player in the coalition game}\} .$$

We verify that our choice of $\mathit{Sat}(\varphi')$ is correct as follows. For all $s \in \mathit{Sat}(\varphi')$, we construct an MDP from the appropriate coalition game by fixing the decisions of the first player according to $c(i, s)$, and in polynomial time we check that the minimal probability (or reward) in the resulting MDP exceeds the bound given by the outermost operator of φ' (see Theorem 2). If $s \notin \mathit{Sat}(\varphi')$, then we fix the decisions of the second player according to $c(i, s)$ and proceed analogously, computing the maximal probabilities.

Theorem 1(b). Model checking an arbitrary rPATL formula is in NEXP ∩ coNEXP.

Proof The proof is similar to that for Theorem 1(a) above. We only need to extend the certificate from the proof to provide a witnessing strategy for formulae of the form $\langle\langle C\rangle\rangle \mathsf{R}^r_{\bowtie x}[\mathsf{F}^0 \phi]$. This is straightforward since, in the proof of Lemma 3, we showed that players need only strategies of exponential size.

In Lemma 3 we have shown that, for the optimal strategy, it suffices to play a deterministic memoryless strategy after a certain reward bound B has been reached and, before that, the strategy needs to remember only the reward accumulated along the history. The rewards are integers, therefore, the strategy in a state may need a different action for each value of reward below B, and one action for reward which is greater or equal to B. So, the overall size of the memory will be $\mathcal{O}(|S| \times B)$. A deterministic strategy suffices in this case; observe that one could ‘embed’ the memory into the game by constructing a new game where the set of states is $S \times \{0, \ldots, B + r_{\max} - 1\} \cup \{s_f\}$. The transition relation is preserved for states (s, k) where $k < B$, and states (s, k) where $k \geq B$ have a transition to s_f only. The reward structure r assigns reward $R_{(s,k)}$ to states where $k \geq B$, which can be computed using step 2 of the algorithm for $\star = 0$ in Section 4, and 0 to all other states. Then, the deterministic memoryless strategy that maximises $\mathit{rew}(r, c, \{s_f\})$ in this new game will also be an optimal strategy in the original game (but requiring memory of size B). The size of B can be at most exponential in the size of $\mathcal{G}$, i.e. from Lemma 3 it follows that the size of B for a state s is bounded by

$$B_s = \frac{\mathbb{E}_{\mathcal{G}_C,s}^{\max,\min}[\mathit{rew}(r, c, T)]}{\Pr_{\mathcal{G}_C,s}^{\max,\min}(\mathsf{F}\, t) - \mathit{PR}_s^{\underline{\max}}} . \tag{7}$$

We claim that all $\mathbb{E}_{\mathcal{G}_C,s}^{\max,\min}[\mathit{rew}(r, c, T)]$, $\Pr_{\mathcal{G}_C,s}^{\max,\min}(\mathsf{F}\, t)$ and $\mathit{PR}_s^{\underline{\max}}$ can be represented as fractions of integers whose binary representation is polynomial in the size of the input, from which the bound on the size of B_s follows. For $\mathbb{E}_{\mathcal{G}_C,s}^{\max,\min}[\mathit{rew}(r, c, T)]$ (or $\Pr_{\mathcal{G}_C,s}^{\max,\min}(\mathsf{F}\, t)$, $\mathit{PR}_s^{\underline{\max}}$), fixing the optimal strategies for both players we can construct a linear program whose size is polynomial in the size of input, and whose solution is equal to $\mathbb{E}_{\mathcal{G}_C,s}^{\max,\min}[\mathit{rew}(r, c, T)]$ (or $\Pr_{\mathcal{G}_C,s}^{\max,\min}(\mathsf{F}\, t)$, $\mathit{PR}_s^{\underline{\max}}$, respectively). Because the solution of the linear program can be represented as a fraction of two integers of polynomial binary representations, we get the claim. Therefore, B_s is at most exponential in the size of $\mathcal{G}$.

D Proof of Correctness for Section 4.4

Price-bounded coalitions. In the proof of Theorem 1 (when all coalitions were fixed), we exploited the fact that there is an exponential size certificate c, which is a function that assigns an element of $\Sigma_1^{C_i} \cup \Sigma_2^{C_i}$ to each tuple (i, s) where $s \in S$ and φ_i is a formula whose outermost operator is temporal.

We extend this approach by changing the certificate c so that it returns an element of $\Sigma_1^C \cup \Sigma_2^C$ to each tuple (i, s, C), where $s \in S$, φ_i is a formula whose outermost operator is temporal, $C \subseteq \Pi$, and $\bowtie \in \{>, \geq\}$:

- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$ or $\varphi_i \equiv \langle\langle ?\rangle\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi]$, and $s \models \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$, then: $c(i, s, C) = \sigma_1$ for $\sigma_1 \in \Sigma_1^C$ such that $\min_{\sigma_2 \in \Sigma_2} \mathrm{Pr}_{\mathcal{G}_C, s}^{\sigma_1, \sigma_2}(\psi) \bowtie q$ holds.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$ or $\varphi_i \equiv \langle\langle ?\rangle\rangle_{\leq y} \mathsf{P}_{\bowtie q}[\psi]$, and $s \not\models \langle\langle C\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$, then: $c(i, s, C) = \sigma_2$ for $\sigma_2 \in \Sigma_2^C$ such that $\max_{\sigma_1 \in \Sigma_1} \mathrm{Pr}_{\mathcal{G}_C, s}^{\sigma_1, \sigma_2}(\psi) \bowtie q$ does *not* hold.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$ or $\varphi_i \equiv \langle\langle ?\rangle\rangle_{\leq y} \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, and $s \models \langle\langle C\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, then: $c(i, s, C) = \sigma_1$ for $\sigma_1 \in \Sigma_1^C$ such that $\min_{\sigma_2 \in \Sigma_2} \mathbb{E}_{\mathcal{G}_C, s}^{\sigma_1, \sigma_2}[\mathit{rew}(r, \star, \mathit{Sat}(\phi))] \bowtie x$ holds.
- If $\varphi_i \equiv \langle\langle C\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$ or $\varphi_i \equiv \langle\langle ?\rangle\rangle_{\leq y} \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, and $s \not\models \langle\langle C\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, then: $c(i, s, C) {=} \sigma_2$ for $\sigma_2 {\in} \Sigma_2^C$ such that $\max_{\sigma_1 \in \Sigma_1} \mathbb{E}_{\mathcal{G}_C, s}^{\sigma_1, \sigma_2}[\mathit{rew}(r, \star, \mathit{Sat}(\phi))] \bowtie x$ does *not* hold.
- If $\varphi_i \equiv \langle\langle C'\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$ $\varphi_i \equiv \langle\langle C'\rangle\rangle \mathsf{P}_{\bowtie q}[\psi]$, $\varphi_i \equiv \langle\langle C'\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, or $\varphi_i \equiv \langle\langle C'\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^\star \phi]$, and $C' \neq C$, then $c(i, s, C)$ returns an arbitrary memoryless deterministic strategy from $\Sigma_1^{C'} \cup \Sigma_2^{C'}$.

As before, the existence of strategies assigned by c follows from Theorem 2 and from Appendix A.2: for all formulae but $\langle\langle C\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^0 \phi]$ and $\langle\langle ?\rangle\rangle \mathsf{R}_{\bowtie x}^r[\mathsf{F}^0 \phi]$, memoryless deterministic strategies exist; and, for the aforementioned formulae, exponential memory deterministic strategies suffice.

To check the certificate in polynomial time (in the worst-case size of c, which is exponential in the size of the model), we compute $\mathit{Sat}(\varphi')$ for all state subformulae φ' of φ, traversing the parse tree of φ bottom-up. Suppose that we are analysing a formula φ' and that we have computed $\mathit{Sat}(\varphi'')$ for all state subformulae φ'' of φ'. If φ' is an atomic proposition or its outermost operator is a boolean connective, we construct $\mathit{Sat}(\varphi')$ in the obvious way. Otherwise, if the outermost coalition in φ' is fixed to C:

$$\mathit{Sat}(\varphi') = \{s \mid c(i, s, C) \in \Sigma_1^C\}$$

and, if the outermost coalition in φ' is not specified, but a coalition of price $\leq y$ is required:

$$\mathit{Sat}(\varphi') = \{s \mid \exists C \subseteq \Pi : \sum_{\gamma \in C} p(\gamma) \leq y \text{ and } c(i, s, C) \in \Sigma_1^C\}\,.$$

We verify that our choice of $\mathit{Sat}(\varphi')$ is correct as follows. For all $s \in \mathit{Sat}(\varphi')$, we construct an MDP from the appropriate coalition game by fixing the decisions of the first player according to $c(i, s, C)$ and in polynomial time we check that the minimal probability (or reward) in the resulting MDP exceeds the bound given by the outermost operator of φ' (see Theorem 2). If $s \notin \mathit{Sat}(\varphi')$, and the outermost coalition of φ' is C, then we fix the decisions of the second player according to $c(i, s, C)$ and proceed analogously, computing the maximal probabilities. If $s \notin \mathit{Sat}(\varphi')$, and the outermost coalition of φ' is not specified, but required to be of price at most y, we need to construct MDPs from the coalition games $\mathcal{G}_C$ for all C where $\sum_{\gamma \in C} p(\gamma) \leq y$ by fixing the decisions of the second player and computing the maximal probabilities. There are only polynomially many (in the size of c) possible choices of C (the number of different coalitions is exponential in the size of $\mathcal{G}$, but the certificate is exponential in $\mathcal{G}$ too), and each choice can be checked in polynomial time.

References

1. M. Aizatulin, H. Schnoor, and T. Wilke. Computationally sound analysis of a probabilistic contract signing protocol. In *Proc. 14th European Symposium on Research in Computer Security (ESORICS'09)*, volume 5789 of *LNCS*, pages 571–586. Springer, 2009.

2. R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
3. R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. 10th International Conference on Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 521–525, Vancouver, 1998. Springer.
4. S. Andova, H. Hermanns, and J.-P. Katoen. Discrete-time rewards model-checked. In *Proc. Formal Modeling and Analysis of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 88–104. Springer, 2003.
5. C. Baier, T. Brázdil, M. Größer, and A. Kucera. Stochastic game logic. In *Proc. 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, pages 227–236. IEEE, 2007.
6. C. Baier and J.-P. Katoen. *Principles of Model Checking.* MIT Press, 2008.
7. P. Ballarini, M. Fisher, and M. Wooldridge. Uncertain agent verification through probabilistic model-checking. In *Proc. 3rd International Workshop on Safety and Security in Multi-agent Systems (SASEMAS'06)*, 2006.
8. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. Thiagarajan, editor, *Proc. 15th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95)*, volume 1026 of *LNCS*, pages 499–513. Springer, 1995.
9. T. Brihaye, N. Markey, M. Ghannem, and L. Rieg. Good friends are hard to find! In S. Demri and C. Jensen, editors, *Proc. 15th International Symposium on Temporal Representation and Reasoning (TIME'08)*, pages 32–40. IEEE, 2008.
10. N. Bulling and W. Jamroga. What agents can probably enforce. *Fundamenta Informaticae*, 93(1–3):81–96, 2009.
11. P. Cerný, K. Chatterjee, T. Henzinger, A. Radhakrishna, and R. Singh. Quantitative synthesis for concurrent programs. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 243–259. Springer, 2011.
12. K. Chatterjee. *Stochastic ω-Regular Games.* PhD thesis, University of California at Berkeley, 2007.
13. K. Chatterjee and T. Henzinger. Value iteration. *25 Years of Model Checking*, pages 107–138, 2008.
14. K. Chatterjee, T. Henzinger, B. Jobstmann, and A. Radhakrishna. Gist: A solver for probabilistic games. In *Proc. 22nd International Conference on Computer Aided Verification (CAV'10)*, LNCS, pages 665–669. Springer, 2010.
15. K. Chatterjee, M. Jurdzinski, and T. Henzinger. Quantitative stochastic parity games. In J. Ian Munro, editor, *Proc. 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 121–130. SIAM, 2004.
16. T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. In C. Flanagan and B. König, editors, *Proc. 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12)*, volume 7214 of *LNCS*, pages 315–330. Springer, 2012.
17. T. Chen, M. Kwiatkowska, D. Parker, and A. Simaitis. Verifying team formation protocols with probabilistic model checking. In *Proc. 12th International Workshop on Computational Logic in Multi-Agent Systems (CLIMA XII 2011)*, volume 6814 of *LNCS*, pages 190–297. Springer, 2011.
18. T. Chen and J. Lu. Probabilistic alternating-time temporal logic and model checking algorithm. In *Proc. 4th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'07)*, pages 35–39. IEEE, 2007.
19. A. Condon. On algorithms for simple stochastic games. *Advances in computational complexity theory, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 13:51–73, 1993.
20. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
21. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In J. Baeten and S. Mauw, editors, *Proc. 10th International Conference on Concurrency Theory (CONCUR'99)*, volume 1664 of *LNCS*, pages 66–81. Springer, 1999.
22. L. de Alfaro and T. Henzinger. Concurrent omega-regular games. In *Proc. 15th Annual IEEE Symposium on Logic in Computer Science*, pages 141–154. IEEE Computer Society, 2000.
23. J. Filar and K. Vrieze. *Competitive Markov Decision Processes.* Springer, 1997.

24. V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In M. Bernardo and V. Issarny, editors, *Formal Methods for Eternal Networked Software Systems (SFM'11)*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.
25. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
26. H. Hildmann and F. Saffre. Influence of variable supply and load flexibility on demand-side management. In *Proc. 8th International Conference on the European Energy Market (EEM'11)*, pages 63–68, 2011.
27. S. Kremer and J.-F. Raskin. A game-based verification of non-repudiation and fair exchange protocols. *Journal of Computer Security*, 11(3):399–430, 2003.
28. M. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In G. Gopalakrishnan and S. Qadeer, editors, *Proc. 23rd International Conference on Computer Aided Verification (CAV'11)*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.
29. F. Laroussinie, N. Markey, and G. Oreiby. On the expressiveness and complexity of ATL. In H. Seidl, editor, *Proc. 10th International Conference on Foundations of Software Science and Computational Structures (FOSSACS'07)*, volume 4423 of *LNCS*, pages 243–257. Springer, 2007.
30. A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proc. 21st International Conference on Computer Aided Verification (CAV'09)*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.
31. D. Martin. The determinacy of Blackwell games. *The Journal of Symbolic Logic*, 63(4):1565–1581, 1998.
32. A. McIver and C. Morgan. Results on the quantitative mu-calculus qMu. *ACM Transactions on Computational Logic*, 8(1), 2007.
33. F. Saffre and A. Simaitis. Host selection through collective decision. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 7(1), 2012.
34. H. Schnoor. Strategic planning for probabilistic games with incomplete information. In *Proc. 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 1057–1064, 2010.
35. M. Ummels. *Stochastic Multiplayer Games: Theory and Algorithms.* PhD thesis, RWTH Aachen University, 2010.
36. W. van der Hoek and M. Wooldridge. Model checking cooperation, knowledge, and time - A case study. *Research In Economics*, 57(3):235–265, 2003.
37. C. Zhang and J. Pang. On probabilistic alternating simulations. In C. Calude and V. Sassone, editors, *Proc. 6th IFIP Conference on Theoretical Computer Science (TCS'10)*, volume 323 of *IFIP*, pages 71–85. Springer, 2010.
38. C. Zhang and J. Pang. An algorithm for probabilistic alternating simulation. In M. Bieliková, G. Friedrich, G. Gottlob, S. Katzenbeisser, and G. Turán, editors, *Proc. 38th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'12)*, volume 7147 of *LNCS*, pages 431–442. Springer, 2012.