Noname manuscript No.
(will be inserted by the editor)

Automatic Verification of Concurrent Stochastic Systems

Marta Kwiatkowska · Gethin Norman · David Parker · Gabriel Santos

the date of receipt and acceptance should be inserted later

Abstract Automated verification techniques for stochastic games allow formal reasoning about systems that feature competitive or collaborative behaviour among rational agents in uncertain or probabilistic settings. Existing tools and techniques focus on turn-based games, where each state of the game is controlled by a single player, and on zero-sum properties, where two players or coalitions have directly opposing objectives. In this paper, we present automated verification techniques for concurrent stochastic games (CSGs), which provide a more natural model of concurrent decision making and interaction. We also consider (social welfare) Nash equilibria, to formally identify scenarios where two players or coalitions with distinct goals can collaborate to optimise their joint performance. We propose an extension of the temporal logic rPATL for specifying quantitative properties in this setting and present corresponding algorithms for verification and strategy synthesis for a variant of stopping games. For finite-horizon properties the computation is exact, while for infinite-horizon it is approximate using value iteration. For zero-sum properties it requires solving matrix games via linear programming, and for equilibria-based properties we find social welfare or social cost Nash equilibria of bimatrix games via the method of labelled polytopes through an SMT encoding. We implement this approach in PRISM-games, which required extending the tool's modelling language for CSGs, and apply it to case studies from domains including robotics, computer security and computer networks, explicitly demonstrating the benefits of both CSGs and equilibria-based properties.

Keywords Quantitative verification · Probabilistic model checking · Concurrent stochastic games · Nash equilibria

Marta Kwiatkowska · Gabriel Santos
Department of Computing Science, University of Oxford, UK

Gethin Norman
School of Computing Science, University of Glasgow, UK

David Parker
School of Computer Science, University of Birmingham, UK

1 Introduction

Stochastic multi-player games are a versatile modelling framework for systems that exhibit cooperative or competitive behaviour in the presence of adversarial or uncertain environments. They can be viewed as a collection of players (agents) with strategies for determining their actions based on the execution so far. These models combine *nondeterminism*, representing the adversarial, cooperative and competitive choices, *stochasticity*, modelling uncertainty due to noise, failures or randomness, and *concurrency*, representing simultaneous execution of interacting agents. Examples of such systems appear in many domains, from robotics and autonomous transport, to security and computer networks. A game-theoretic approach also facilitates the design of protocols that use penalties or incentives to ensure robustness against selfish participants. However, the complex interactions involved in such systems make their correct construction a challenge.

Formal verification for stochastic games provides a means of producing quantitative guarantees on the correctness of these systems (e.g. "the control software can always safely stop the vehicle with probability at least 0.99, regardless of the actions of other road users"), where the required behavioural properties are specified precisely in quantitative extensions of temporal logic. The closely related problem of *strategy synthesis* constructs an optimal strategy for a player, or coalition of players, which guarantees that such a property is satisfied.

A variety of verification algorithms for stochastic games have been devised, e.g., [17,77,3,4,18]. In recent years, further progress has been made: verification and strategy synthesis algorithms have been developed for various temporal logics [23,9,45,42] and implemented in the PRISM-games tool [51], an extension of the PRISM probabilistic model checker [44]. This has allowed modelling and verification of stochastic games to be used for a variety of non-trivial applications, in which competitive or collaborative behaviour between entities is a crucial ingredient, including computer security and energy management.

A limitation of the techniques implemented in PRISM-games to date is that they focus on *turn-based* stochastic multi-player games (TSGs), whose states are partitioned among a set of players, with exactly one player taking control of each state. In this paper, we propose and implement techniques for *concurrent stochastic multi-player games* (CSGs), which generalise TSGs by permitting players to choose their actions simultaneously in each state. This provides a more realistic model of interactive agents operating concurrently, and making action choices without already knowing the actions being taken by other agents. Although algorithms for CSGs have been known for some time (e.g., [3,4,18]), their implementation and application to real-world examples has been lacking.

A further limitation of existing work is that it focuses on *zero-sum* properties, in which one player (or a coalition of players) aims to optimise some objective, while the remaining players have the directly opposing goal. In PRISM-games, properties are specified in the logic rPATL (probabilistic alternating-time temporal logic with rewards) [23], a quantitative extension of the game logic ATL [5]. This allows us to specify that a coalition of players can achieve a high-level objective, regarding the probability of an event's occurrence or the expectation of reward measure, irrespective of the other players' strategies. Extensions have allowed players to optimise multiple objectives [24,9], but again in a zero-sum fashion.

In this work, we move beyond zero-sum properties and consider situations where two players (or two coalitions of players) in a CSG have distinct objectives to be maximised or minimised. The goals of the players (or coalitions) are not necessarily directly opposing, and so it may be beneficial for players to collaborate. For these *nonzero-sum* scenarios, we use the well studied notion of *Nash equilibria* (NE), where it is not beneficial for any player to unilaterally change their strategy. In particular, we use subgame-perfect NE [63], where this equilibrium criterion holds in every state of the game, and we focus on two specific variants of equilibria: *social welfare* and *social cost* NE, which maximise and minimise, respectively, the sum of the objectives of the players.

We propose an extension of the rPATL logic which adds the ability to express quantitative *nonzero-sum* properties based on these notions of equilibria, for example "the two robots have navigation strategies which form a (social cost) Nash equilibrium, and under which the combined expected energy usage until completing their tasks is below k". We also include some additional reward properties that have proved to be useful when applying our methods to various case studies.

We provide a formal semantics for the new logic and propose algorithms for CSG verification and strategy synthesis for a variant of stopping games, including both zero-sum and nonzero-sum properties. Our algorithms extend the existing approaches for rPATL model checking, and employ a combination of exact computation through backward induction for finite-horizon properties and approximate computation through value iteration for infinite-horizon properties. Both approaches require the solution of games for each state of the model in each iteration of the computation: we solve matrix games for the zero-sum case and find optimal NE for bimatrix games for the nonzero-sum case. The former can be done with linear programming; we perform the latter using labelled polytopes [52] and a reduction to SMT.

We have implemented our verification and strategy synthesis algorithms in a new release, version 3.0, of PRISM-games [48], extending both the modelling and property specification languages to support CSGs and nonzero-sum properties. In order to investigate the performance, scalability and applicability of our techniques, we have developed a large selection of case studies taken from a diverse set of application domains including: finance, computer security, computer networks, communication systems, robot navigation and power control.

These illustrate examples of systems whose modelling and analysis requires stochasticity *and* competitive or collaborative behaviour between concurrent components or agents. We demonstrate that our CSG modelling and verification techniques facilitate insightful analysis of quantitative aspects of such systems. Specifically, we show cases where CSGs allow more accurate modelling of concurrent behaviour than their turn-based counterparts and where our equilibria-based extension of rPATL allows us to synthesise better performing strategies for collaborating agents than can be achieved using the zero-sum version.

The paper combines and extends the conference papers [45] and [46]. In particular, we: (i) introduce the definition of social cost Nash equilibria for CSGs and model checking algorithms for verifying temporal logic specifications using this definition; (ii) provide additional details and proofs of model checking algorithms, for example for combinations of finite- and infinite-horizon objectives; (iii) present an expanded experimental evaluation, including a wider range of properties, extended analysis of the case studies and a more detailed evaluation of performance, including efficiency improvements with respect to [45,46].

Related work. Various verification algorithms have been proposed for CSGs, e.g. [3,4,18], but without implementations, tool support or case studies. PRISM-games 2.0 [51], which we have built upon in this work, provided modelling and verification for a wide range of properties of stochastic multi-player games, including those in the logic rPATL, and multi-objective extensions of it, but focusing purely on the turn-based variant of the model (TSGs) in the context of two-coalitional zero-sum properties. GIST [21] allows the analysis of ω-regular properties on probabilistic games, but again focuses on turn-based, not concurrent, games. GAVS+ [25] is a general-purpose tool for algorithmic game solving, supporting TSGs and (non-stochastic) concurrent games, but not CSGs. Three further tools, PRALINE [14], EAGLE [76] and EVE [34], support the computation of NE [58] for the restricted class of (non-stochastic) concurrent games. In addition, EVE has recently been extended to verify if an LTL property holds on some or all NE [35]. Computing NE is also supported by MCMAS-SLK [16] via strategy logic and general purpose tools such as Gambit [57] can compute a variety of equilibria but, again, not for stochastic games.

Work concerning nonzero-sum properties includes [22,77], in which the existence of and the complexity of finding NE for stochastic games is studied, but without practical algorithms. The complexity of finding subgame-perfect NE for quantitative reachability properties is studied in [15], while [33] considers the complexity of equilibrium design for temporal logic properties and lists social welfare requirements and implementation as future work. In [67], a learning-based algorithm for finding NE for discounted properties of CSGs is presented and evaluated. Similarly, [53] studies NE for discounted properties and introduces iterative algorithms for strategy synthesis. A theoretical framework for price-taking equilibria of CSGs is given in [6], where players try to minimise their costs which include a price common to all players and dependent on the decisions of all players. A notion of strong NE for a restricted class of CSGs is formalised in [27] and an approximation algorithm for checking the existence of such NE for discounted properties is introduced and evaluated. The existence of stochastic equilibria with imprecise deviations for CSGs and a PSPACE algorithm to compute such equilibria is considered in [12]. Finally, we mention the fact that the concept of equilibrium is used to analyze different applications such as cooperation among agents in stochastic games [39] and to design protocols based on quantum secret sharing [69].

2 Preliminaries

We begin with some basic background from game theory, and then describe CSGs, illustrating each with examples. For any finite set X, we will write $\mathit{Dist}(X)$ for the set of probability distributions over X and for any vector $v \in \mathbb{Q}^n$ for $n \in \mathbb{N}$ we use $v(i)$ to denote the ith entry of the vector.

2.1 Game Theory Concepts

We first introduce *normal form games*, which are simple one-shot games where players make their choices concurrently.

Definition 1 (Normal form game) A (finite, n-person) *normal form game* (NFG) is a tuple $\mathsf{N} = (N, A, u)$ where:

- $N = \{1, \dots, n\}$ is a finite set of *players*;
- $A = A_1 \times \cdots \times A_n$ and A_i is a finite set of *actions* available to player $i \in N$;
- $u = (u_1, \dots, u_n)$ and $u_i \colon A \to \mathbb{Q}$ is a *utility function* for player $i \in N$.

In a game N, players select actions simultaneously, with player $i \in N$ choosing from the action set A_i. If each player i selects action a_i, then player j receives the utility $u_j(a_1, \dots, a_n)$.

Definition 2 (Strategies and strategy profile) A *(mixed) strategy* σ_i for player i in an NFG N is a distribution over its action set, i.e., $\sigma_i \in \mathit{Dist}(A_i)$. We let Σ^i_{N} denote the set of all strategies for player i. A *strategy profile* (or just *profile*) $\sigma = (\sigma_1, \dots, \sigma_n)$ is a tuple of strategies for each player.

Under a strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ of an NFG N, the expected utility of player i is defined as follows:

$$u_i(\sigma) \ \stackrel{\text{def}}{=} \ \textstyle\sum_{(a_1,\dots,a_n)\in A} u_i(a_1, \dots, a_n) \cdot \left(\prod_{j=1}^n \sigma_j(a_j) \right) .$$

A two-player NFG is also called a *bimatrix game* as it can be represented by two distinct matrices $\mathsf{Z}_1, \mathsf{Z}_2 \in \mathbb{Q}^{l \times m}$ where $A_1 = \{a_1, \dots, a_l\}$, $A_2 = \{b_1, \dots, b_m\}$, $z^1_{ij} = u_1(a_i, b_j)$ and $z^2_{ij} = u_2(a_i, b_j)$.

A two-player NFG is *constant-sum* if there exists $c \in \mathbb{Q}$ such that $u_1(\alpha) + u_2(\alpha) = c$ for all $\alpha \in A$ and *zero-sum* if $c = 0$. A zero-sum, two-player NFG is often called a *matrix game* as it can be represented by a single matrix $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ where $A_1 = \{a_1, \dots, a_l\}$, $A_2 = \{b_1, \dots, b_m\}$ and $z_{ij} = u_1(a_i, b_j) = -u_2(a_i, b_j)$. For zero-sum, two-player NFGs, in the bimatrix game representation we have $\mathsf{Z}_1 = -\mathsf{Z}_2$.

2.1.1 Matrix Games

We require the following classical result concerning matrix games, which introduces the notion of the *value* of a matrix game (and zero-sum NFG).

Theorem 1 (Minimax theorem [59, 60]) *For any zero-sum NFG $\mathsf{N} = (N, A, u)$ and corresponding matrix game Z, there exists $v^\star \in \mathbb{Q}$, called the* value *of the game and denoted $\mathit{val}(\mathsf{Z})$, such that:*

- *there is a strategy $\sigma_1^\star$ for player 1, called an optimal strategy of player 1, such that under this strategy the player's expected utility is at least $v^\star$ regardless of the strategy of player 2, i.e. $\inf_{\sigma_2 \in \Sigma^2_{\mathsf{N}}} u_1(\sigma_1^\star, \sigma_2) \geqslant v^\star$;*
- *there is a strategy $\sigma_2^\star$ for player 2, called an optimal strategy of player 2, such that under this strategy the player's expected utility is at least $-v^\star$ regardless of the strategy of player 1, i.e. $\inf_{\sigma_1 \in \Sigma^1_{\mathsf{N}}} u_2(\sigma_1, \sigma_2^\star) \geqslant -v^\star$.*

The value of a matrix game $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ can be found by solving the following linear programming (LP) problem [59,60]. Maximise v subject to the constraints:

$$\begin{aligned} x_1 {\cdot} z_{1j} + \cdots + x_l {\cdot} z_{lj} &\geqslant v \;\; \text{for all } 1 \leqslant j \leqslant m \\ x_i &\geqslant 0 \;\; \text{for all } 1 \leqslant i \leqslant l \\ x_1 + \cdots + x_l &= 1 \end{aligned}$$

In addition, the solution for $(x_1, \ldots, x_l)$ yields an optimal strategy for player 1. The value of the game can also be found by solving the following dual LP problem. Minimise v subject to the constraints:

$$\begin{aligned} y_1 {\cdot} z_{i1} + \cdots + y_m {\cdot} z_{im} &\leqslant v \;\; \text{for all } 1 \leqslant i \leqslant l \\ y_j &\geqslant 0 \;\; \text{for all } 1 \leqslant j \leqslant m \\ y_1 + \cdots + y_m &= 1 \end{aligned}$$

and in this case the solution $(y_1, \ldots, y_m)$ yields an optimal strategy for player 2.

Example 1. Consider the (zero-sum) NFG corresponding to the well known rock-paper-scissors game, where each player $i \in \{1, 2\}$ chooses "rock" (r_i), "paper" (p_i) or "scissors" (s_i). The matrix game representation is:

$$\mathsf{Z} = \begin{array}{c} \\ r_1 \\ p_1 \\ s_1 \end{array} \begin{array}{c} \begin{array}{ccc} r_2 & p_2 & s_2 \end{array} \\ \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix} \end{array}$$

where the utilities for winning, losing and drawing are 1, -1 and 0 respectively. The value for this matrix game is the solution to the following LP problem. Maximise v subject to the constraints:

$$\begin{aligned} x_2 - x_3 &\geqslant v \\ x_3 - x_1 &\geqslant v \\ x_1 - x_2 &\geqslant v \\ x_1, x_2, x_3 &\geqslant 0 \\ x_1 + x_2 + x_3 &= 1 \end{aligned}$$

which yields the value $v^\star = 0$ with optimal strategy $\sigma_1^\star = (1/3, 1/3, 1/3)$ for player 1 (the optimal strategy for player 2 is the same). ■

2.1.2 Bimatrix Games

For bimatrix games (and nonzero-sum NFGs), we use the concept of *Nash equilibria* (NE), which represent scenarios for players with distinct objectives in which it is not beneficial for any player to unilaterally change their strategy. In particular, we will use variants called *social welfare optimal* NE and *social cost optimal* NE. These variants are equilibria that maximise or minimise, respectively, the total utility of the players, i.e., the sum of the individual player utilities.

Definition 3 (Best and least response) For NFG $\mathsf{N} = (N, A, u)$, strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ and player i strategy σ'_i, we define the sequence of strategies $\sigma_{-i} = (\sigma_1, \dots, \sigma_{i-1}, \sigma_{i+1}, \dots, \sigma_n)$ and profile $\sigma_{-i}[\sigma'_i] = (\sigma_1, \dots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \dots, \sigma_n)$. For player i and strategy sequence σ_{-i}:

– a *best response* for player i to σ_{-i} is a strategy $\sigma^\star_i$ for player i such that $u_i(\sigma_{-i}[\sigma^\star_i]) \geqslant u_i(\sigma_{-i}[\sigma_i])$ for all strategies σ_i of player i;
– a *least response* for player i to σ_{-i} is a strategy $\sigma^\star_i$ for player i such that $u_i(\sigma_{-i}[\sigma^\star_i]) \leqslant u_i(\sigma_{-i}[\sigma_i])$ for all strategies σ_i of player i.

Definition 4 (Nash equilibrium) For NFG $\mathsf{N} = (N, A, u)$, a strategy profile $\sigma^\star$ of N is a *Nash equilibrium* (NE) and $\langle u_i(\sigma^\star)\rangle_{i \in N}$ *NE values* if $\sigma^\star_i$ is a best response to $\sigma^\star_{-i}$ for all $i \in N$.

Definition 5 (Social welfare NE) For NFG $\mathsf{N} = (N, A, u)$, an NE $\sigma^\star$ of N is a *social welfare optimal NE* (SWNE) and $\langle u_i(\sigma^\star)\rangle_{i \in N}$ corresponding *SWNE values* if $u_1(\sigma^\star)+\cdots+u_n(\sigma^\star) \geqslant u_1(\sigma)+\cdots+u_n(\sigma)$ for all NE σ of N.

Definition 6 (Social cost NE) For NFG $\mathsf{N} = (N, A, u)$, a profile $\sigma^\star$ of N is a *social cost optimal NE* (SCNE) and $\langle u_i(\sigma^\star)\rangle_{i \in N}$ corresponding *SCNE values* if it is an NE of $\mathsf{N}^- = (N, A, -u)$ and $u_1(\sigma^\star)+\cdots+u_n(\sigma^\star) \leqslant u_1(\sigma)+\cdots+u_n(\sigma)$ for all NE σ of $\mathsf{N}^- = (N, A, -u)$.

The notion of SWNE is standard [61] and corresponds to the case where utility values represent profits or rewards. We introduce the dual notion of SCNE for the case where utility values correspond to losses or costs. In our experience of modelling with stochastic games, such situations are common: example objectives in this category include minimising the probability of a fault occurring or minimising the expected time to complete a task. Representing SCNE directly is a more natural approach than the alternative of simply negating utilities, as above.

The following demonstrates the relationship between SWNE and SCNE.

Lemma 1 *For NFG $\mathsf{N} = (N, A, u)$, a strategy profile $\sigma^\star$ of N is an NE of $\mathsf{N}^- = (N, A, -u)$ if and only if $\sigma^\star_i$ is a least response to $\sigma^\star_{-i}$ of player i in N for all $i \in N$. Furthermore, $\sigma^\star$ is a SWNE of N^- if and only if $\sigma^\star$ is a SCNE of N.*

Lemma 1 can be used to reduce the computation of SCNE profiles and values to those of SWNE profiles and values (or vice versa). This is achieved by negating all utilities in the NFG or bimatrix game, computing an SWNE profile and corresponding SWNE values, and then negating the SWNE values to obtain an SCNE profile and corresponding SCNE values for the original NFG or bimatrix game.

Finding NE and NE values in bimatrix games is in the class of *linear complementarity* problems (LCPs). More precisely, (σ_1, σ_2) is an NE profile and (u, v) are the corresponding NE values of the bimatrix game $\mathsf{Z}_1, \mathsf{Z}_2 \in \mathbb{Q}^{l \times m}$ where $A_1 = \{a_1, \dots, a_l\}$, $A_2 = \{b_1, \dots, b_m\}$ if and only if for the column vectors $x \in \mathbb{Q}^l$ and $y \in \mathbb{Q}^m$ where $x_i = \sigma_1(a_i)$ and $y_j = \sigma_2(b_j)$ for $1 \leqslant i \leqslant l$ and $1 \leqslant j \leqslant m$, we have:

$$x^T(\mathbf{1}u - \mathsf{Z}_1 y) = 0 \tag{1}$$
$$y^T(\mathbf{1}v - \mathsf{Z}_2^T x) = 0 \tag{2}$$
$$\mathbf{1}u - \mathsf{Z}_1 y \geqslant \mathbf{0} \tag{3}$$
$$\mathbf{1}v - \mathsf{Z}_2^T x \geqslant \mathbf{0} \tag{4}$$

and **0** and **1** are vectors or matrices with all components 0 and 1, respectively.

Example 2. We consider a nonzero-sum stag hunt game [64] where, if players decide to cooperate, this can yield a large utility, but if the others do not, then the cooperating player gets nothing while the remaining players get a small utility. A scenario with 3 players, where two form a coalition (assuming the role of player 2), yields a bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c} \\ nc_1 \\ c_1 \end{array} \!\!\begin{array}{c} \begin{array}{ccc} nc_2 & hc_2 & c_2 \end{array} \\ \left(\begin{array}{ccc} 2 & 2 & 2 \\ 0 & 4 & 6 \end{array}\right) \end{array} \qquad \mathsf{Z}_2 = \begin{array}{c} \\ nc_1 \\ c_1 \end{array} \!\!\begin{array}{c} \begin{array}{ccc} nc_2 & hc_2 & c_2 \end{array} \\ \left(\begin{array}{ccc} 4 & 2 & 0 \\ 4 & 6 & 9 \end{array}\right) \end{array}$$

where nc_i and c_i represent player 1 and coalition 2 not cooperating and cooperating, respectively, and hc_2 represents half the players in the coalition cooperating. A strategy profile $\sigma^* = ((x_1, x_2, x_3), (y_1, y_2))$ is an NE and (u, v) the corresponding NE values of the game if and only if, from Eqn (1) and (2):

$$u{\cdot}x_1 - 2{\cdot}x_1{\cdot}y_1 - 2{\cdot}x_1{\cdot}y_2 - 2{\cdot}x_1{\cdot}y_3 + u{\cdot}x_2 - 4{\cdot}x_2{\cdot}y_2 - 6{\cdot}x_2{\cdot}y_3 = 0$$
$$v{\cdot}y_1 - 4{\cdot}x_1{\cdot}y_1 - 4{\cdot}x_2{\cdot}y_1 + v{\cdot}y_2 - 2{\cdot}x_1{\cdot}y_2 - 6{\cdot}x2{\cdot}y_2 + v{\cdot}y_3 - 9{\cdot}x_2{\cdot}y_3 = 0$$

and, from Eqn (3) and (4):

$$\begin{aligned} u - 2{\cdot}y_1 - 2{\cdot}y_2 - 2{\cdot}y_3 &\geqslant 0 \\ u - 4{\cdot}y_2 - 6{\cdot}y_3 &\geqslant 0 \\ v - 4{\cdot}x_1 - 4{\cdot}x_2 &\geqslant 0 \\ v - 2{\cdot}x_1 - 6{\cdot}x_2 &\geqslant 0 \\ v - 9{\cdot}x_2 &\geqslant 0\,. \end{aligned}$$

There are three solutions to this LCP problem which correspond to the following NE profiles:

- player 1 and the coalition pick nc_1 and nc_2, respectively, with NE values $(2, 4)$;
- player 1 selects nc_1 and c_1 with probabilities 5/9 and 4/9 and the coalition selects nc_2 and c_2 with probabilities 2/3 and 1/3, with NE values $(2, 4)$;
- player 1 and the coalition select c_1 and c_2, respectively, with NE values $(6, 9)$.

For instance, in the first case, neither player 1 nor the coalition believes the other will cooperate: the best they can do is act alone. The third maximises the joint utility and is the only SWNE profile, with corresponding SWNE values $(6, 9)$.

To find SCNE profiles and SCNE values for the same set of utility functions, using Lemma 1 we can negate all the utilities of the players in the game and look for NE profiles in the resulting bimatrix game; again, there are three:

- player 1 and the coalition select c_1 and nc_2, respectively, with NE values $(0, -4)$;
- player 1 selects nc_1 and c_1 with probabilities 1/2 and 1/2 and the coalition selects nc_2 and hc_2 with probabilities 1/2 and 1/2, with NE values $(-2, -4)$;
- player 1 and the coalition select nc_1 and c_2, respectively, with NE values $(-2, 0)$.

The third is the only SCNE profile, with corresponding SCNE values $(2, 0)$. ■

In this work, we compute the SWNE values for a bimatrix game (or, via Lemma 1, the SCNE values) by first identifying all the NE values of the game. For this, we use the Lemke-Howson algorithm [52], which is based on the method of *labelled polytopes* [61]. Other well-known methods include those based on *support enumeration* [66] and *regret minimisation* [71]. Given a bimatrix game $\mathsf{Z}_1, \mathsf{Z}_2 \in \mathbb{Q}^{l \times m}$, we denote the sets of deterministic strategies of players 1 and 2 by $I = \{1, \dots, l\}$ and $M = \{1, \dots, m\}$ and define $J = \{l{+}1, \dots, l{+}m\}$ by mapping $j \in M$ to $l{+}j \in J$. A *label* is then defined as an element of $I \cup J$. The sets of strategies for players 1 and 2 can be represented by:

$$X = \{x \in \mathbb{Q}^l \mid (\mathbf{1}x = 1) \wedge (x \geqslant \mathbf{0})\} \quad \text{and} \quad Y = \{y \in \mathbb{Q}^m \mid (\mathbf{1}y = 1) \wedge (y \geqslant \mathbf{0})\}\,.$$

The strategy set Y is then divided into regions $Y(i)$ and $Y(j)$ (polytopes) for $i \in I$ and $j \in J$ such that $Y(i)$ contains strategies for which the deterministic strategy i of player 1 is a best response and $Y(j)$ contain strategies which choose action j with probability 0:

$$Y(i) = \{y \in Y \mid \forall k \in I.\ \mathsf{Z}_1(i,:)y \geqslant \mathsf{Z}_1(k,:)y\} \ \text{ and } \ Y(j) = \{y \in Y \mid y_{j-l} = 0\}$$

where $\mathsf{Z}_1(i,:)$ is the ith row vector of Z_1. A vector y is then said to have label k if $y \in Y(k)$, for $k \in I \cup J$. The strategy set X is divided analogously into regions $X(j)$ and $X(i)$ for $j \in J$ and $i \in I$ and a vector x has label k if $x \in X(k)$, for $k \in I \cup J$. A pair of vectors $(x, y) \in X{\times}Y$ is *completely labelled* if the union of the labels of x and y equals $I \cup J$.

The NE profiles of the game are the vector pairs that are completely labelled [52,74]. The corresponding NE values can be computed through matrix-vector multiplication. A SWNE profile and corresponding SWNE values can then be found through an NE profile with NE values that maximise the sum.

2.2 Concurrent Stochastic Games

We now define concurrent stochastic games [73], where players repeatedly make simultaneous choices over actions that update the game state probabilistically.

Definition 7 (Concurrent stochastic game) A *concurrent stochastic multi-player game* (CSG) is a tuple $\mathsf{G} = (N, S, \bar{S}, A, \Delta, \delta, AP, L)$ where:

- $N = \{1, \dots, n\}$ is a finite set of players;
- S is a finite set of states and $\bar{S} \subseteq S$ is a set of initial states;
- $A = (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$ where A_i is a finite set of actions available to player $i \in N$ and $\bot$ is an idle action disjoint from the set $\cup_{i=1}^n A_i$;
- $\Delta \colon S \to 2^{\cup_{i=1}^n A_i}$ is an action assignment function;
- $\delta \colon S{\times}A \to \mathit{Dist}(S)$ is a probabilistic transition function;
- AP is a set of atomic propositions and $L \colon S \to 2^{AP}$ is a labelling function.

A CSG G starts in an initial state $\bar{s} \in \bar{S}$ and, when in state s, each player $i \in N$ selects an action from its available actions $A_i(s) \overset{\text{def}}{=} \Delta(s) \cap A_i$ if this set is non-empty, and from $\{\bot\}$ otherwise. Supposing each player i selects action a_i, the state of the game is updated according to the distribution $\delta(s, (a_1, \dots, a_n))$. A CSG is a *turn-based stochastic multi-player game* (TSG) if for any state s there is precisely one player i for which $A_i(s) \neq \{\bot\}$. Furthermore, a CSG is a *Markov decision process* (MDP) if there is precisely one player i such that $A_i(s) \neq \{\bot\}$ for all states s.

A *path* π of G is a sequence $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots$ where $s_i \in S$, $\alpha_i \in A$ and $\delta(s_i, \alpha_i)(s_{i+1}) > 0$ for all $i \geqslant 0$. We denote by $\pi(i)$ the $(i{+}1)$th state of π, $\pi[i]$ the action associated with the $(i{+}1)$th transition and, if π is finite, $\mathit{last}(\pi)$ the final state. The length of a path π, denoted $|\pi|$, is the number of transitions appearing in the path. Let $\mathit{FPaths}_{\mathsf{G}}$ and $\mathit{IPaths}_{\mathsf{G}}$ ($\mathit{FPaths}_{\mathsf{G},s}$ and $\mathit{IPaths}_{\mathsf{G},s}$) be the sets of finite and infinite paths (starting in state s).

We augment CSGs with *reward structures* of the form $r = (r_A, r_S)$, where $r_A \colon S{\times}A \to \mathbb{Q}$ is an action reward function (which maps each state and action tuple pair to a rational value that is accumulated when the action tuple is selected in the state) and $r_S \colon S \to \mathbb{Q}$ is a state reward function (which maps each state to a rational value that is incurred when the state is reached). We allow both positive and negative rewards; however, we will later impose certain restrictions to ensure the correctness of our model checking algorithms.

A *strategy* for a player in a CSG resolves the player's choices in each state. These choices can depend on the history of the CSG's execution and can be randomised. Formally, we have the following definition.

Definition 8 (Strategy) A *strategy* for player i in a CSG G is a function of the form $\sigma_i \colon \mathit{FPaths}_{\mathsf{G}} \to \mathit{Dist}(A_i \cup \{\bot\})$ such that, if $\sigma_i(\pi)(a_i) > 0$, then $a_i \in A_i(\mathit{last}(\pi))$. We denote by Σ^i_{G} the set of all strategies for player i.

As for NFGs, a *strategy profile* for G is a tuple $\sigma = (\sigma_1, \dots, \sigma_n)$ of strategies for all players and, for player i and strategy σ'_i, we define the sequence σ_{-i} and profile $\sigma_{-i}[\sigma'_i]$ in the same way. For strategy profile $\sigma = (\sigma_1, \dots, \sigma_n)$ and state s, we let $\mathit{FPaths}^{\sigma}_{\mathsf{G},s}$ and $\mathit{IPaths}^{\sigma}_{\mathsf{G},s}$ denote the finite and infinite paths from s under the choices of σ. We can define a probability measure $\mathit{Prob}^{\sigma}_{\mathsf{G},s}$ over the infinite paths $\mathit{IPaths}^{\sigma}_{\mathsf{G},s}$ [43]. This construction is based on first defining the probabilities for finite paths from the probabilistic transition function and choices of the strategies in the profile. More precisely, for a finite path $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{m-1}} s_m$ where $s_0 = s$, the probability of π under the profile σ is defined by:

$$\mathbf{P}^{\sigma}(\pi) \ \stackrel{\text{def}}{=}\ \prod_{j=0}^{m-1} \left(\left(\prod_{i=1}^{n} \sigma_i(s_0 \xrightarrow{\alpha_0} \cdots \xrightarrow{\alpha_{j-1}} s_j)(\alpha_j(i)) \right) \cdot \delta(s_j, \alpha_j)(s_{j+1}) \right).$$

Next, for each finite path π, we define the basic cylinder $C^{\sigma}(\pi)$ that consists of all infinite paths in $\mathit{IPaths}^{\sigma}_{\mathsf{G},s}$ that have π as a prefix. Finally, using properties of cylinders, we can then construct the probability space $(\mathit{IPaths}^{\sigma}_{\mathsf{G},s}, \mathcal{F}^{\sigma}_s, \mathit{Prob}^{\sigma}_{\mathsf{G},s})$, where $\mathcal{F}^{\sigma}_s$ is the smallest σ-algebra generated by the set of basic cylinders $\{C^{\sigma}(\pi) \mid \pi \in \mathit{FPaths}^{\sigma}_{\mathsf{G},s}\}$ and $\mathit{Prob}^{\sigma}_{\mathsf{G},s}$ is the unique measure such that $\mathit{Prob}^{\sigma}_{\mathsf{G},s}(C^{\sigma}(\pi)) = \mathbf{P}^{\sigma}(\pi)$ for all $\pi \in \mathit{FPaths}^{\sigma}_{\mathsf{G},s}$.

For random variable $X \colon \mathit{IPaths}_{\mathsf{G}} \to \mathbb{Q}$, we can then define for any profile σ and state s the expected value $\mathbb{E}^{\sigma}_{\mathsf{G},s}(X)$ of X in s with respect to σ. These random variables X represent an *objective* (or utility function) for a player, which includes both *finite-horizon* and *infinite-horizon* properties. Examples of finite-horizon properties include the probability of reaching a set of target states T within k steps or the expected reward accumulated over k steps. These properties can be expressed by the random variables:

$$X(\pi) = \begin{cases} 1 & \text{if } \pi(j) \in T \text{ for some } j \leqslant k \\ 0 & \text{otherwise} \end{cases}$$

$$Y(\pi) = \textstyle\sum_{i=0}^{k-1} \big(r_A(\pi(i), \pi[i]) + r_S(\pi(i)) \big)$$

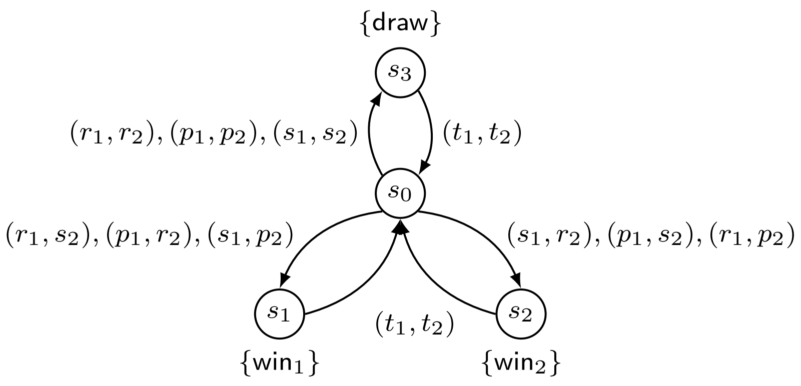

{draw}
s_3
$(r_1,r_2),(p_1,p_2),(s_1,s_2)$
(t_1,t_2)
s_0
$(r_1,s_2),(p_1,r_2),(s_1,p_2)$
$(s_1,r_2),(p_1,s_2),(r_1,p_2)$
s_1
(t_1,t_2)
s_2
{win$_1$}
{win$_2$}

Fig. 1: Rock-paper-scissors CSG.

respectively. Examples of infinite-horizon properties include the probability of reaching a target set T and the expected cumulative reward until reaching a target set T (where paths that never reach the target have infinite reward), which can be expressed by the random variables:

$$X(\pi) = \begin{cases} 1 & \text{if } \pi(j) \in T \text{ for some } j \in \mathbb{N} \\ 0 & \text{otherwise} \end{cases}$$

$$Y(\pi) = \begin{cases} \sum_{i=0}^{k_{\min}-1} \big(r_A(\pi(i), \pi[i]) + r_S(\pi(i))\big) & \text{if } \pi(j) \in T \text{ for some } j \in \mathbb{N} \\ \infty & \text{otherwise} \end{cases}$$

where $k_{\min} = \min\{j \in \mathbb{N} \mid \pi(j) \in T\}$, respectively.

Let us first focus on *zero-sum* games, which are by definition two-player games. As for NFGs (see Definition 1), for a two-player CSG G and a given objective X, we can consider the case where player 1 tries to maximise the expected value of X, while player 2 tries to minimise it. The above definition yields the *value* of G with respect to X if it is determined, i.e., if the maximum value that player 1 can ensure equals the minimum value that player 2 can ensure. Since the CSGs we consider are finite state and finitely-branching, it follows that they are determined for all the objectives we consider [55]. Formally we have the following.

Definition 9 (Determinacy and optimality) For a two-player CSG G and objective X, we say that G is *determined* with respect to X if, for any state s:

$$\sup_{\sigma_1 \in \Sigma^1} \inf_{\sigma_2 \in \Sigma^2} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G},s}(X) \ = \inf_{\sigma_2 \in \Sigma^2} \sup_{\sigma_1 \in \Sigma^1} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G},s}(X)$$

and call this the *value* of G in state s with respect to X, denoted $val_{\mathsf{G}}(s,X)$. Furthermore, a strategy $\sigma_1^\star$ of player 1 is *optimal* with respect to X if we have $\mathbb{E}^{\sigma_1^\star,\sigma_2}_{\mathsf{G},s}(X) \geqslant val_{\mathsf{G}}(s,X)$ for all $s \in S$ and $\sigma_2 \in \Sigma^2$ and a strategy of player 2 is *optimal* with respect to X if $\mathbb{E}^{\sigma_1,\sigma_2^\star}_{\mathsf{G},s}(X) \leqslant val_{\mathsf{G}}(s,X)$ for all $s \in S$ and $\sigma_1 \in \Sigma^1$.

Example 3. Consider the (non-probabilistic) CSG shown in Figure 1 corresponding to two players repeatedly playing the rock-paper-scissors game (see Example 1). Transitions are labelled with action pairs, where $A_i = \{r_i, p_i, s_i, t_i\}$ for $1 \leqslant i \leqslant 2$, with r_i, p_i and s_i representing playing rock, paper and scissors, respectively, and t_i restarting the game. The CSG starts in state s_0 and states s_1, s_2 and s_3 are

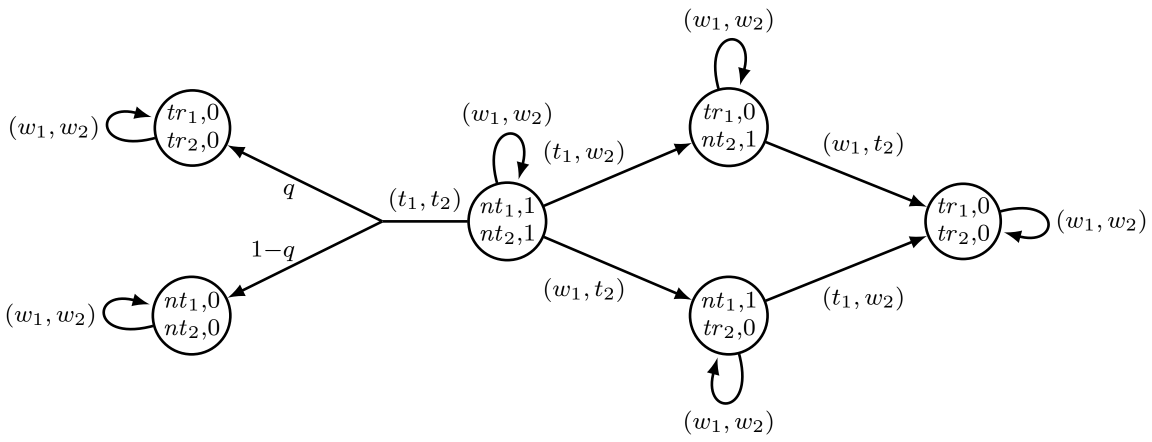


Fig. 2: CSG model of a medium access control problem.

labelled with atomic propositions corresponding to when a player wins or there is a draw in a round of the rock-paper-scissors game.

For the zero-sum objective to maximise the probability of reaching s_1 before s_2, i.e. player 1 winning a round of the game before player 2, the value of the game is 1/2 and the optimal strategy of each player i is to choose r_i, p_i and s_i, each with probability 1/3 in state s_0 and t_i otherwise. ■

For *nonzero-sum* CSGs, with an objective X_i for each player i, we will use NE, which can be defined as for NFGs (see Definition 4). In line with the definition of zero-sum optimality above (and because the model checking algorithms we will later introduce are based on backward induction [72,60]), we restrict our attention to *subgame-perfect* NE [63], which are NE in *every state* of the CSG.

Definition 10 (Subgame-perfect NE) For CSG G, a strategy profile $\sigma^\star$ is a *subgame-perfect Nash equilibrium* for objectives $\langle X_i \rangle_{i \in N}$ if and only if $\mathbb{E}^{\sigma^\star}_{\mathsf{G},s}(X_i) \geqslant \sup_{\sigma_i \in \Sigma_i} \mathbb{E}^{\sigma^\star_{-i}[\sigma_i]}_{\mathsf{G},s}(X_i)$ for all $i \in N$ and $s \in S$.

Furthermore, because we use a variety of objectives, including infinite-horizon objectives, where the existence of NE is an open problem [11], we will in some cases use ε-NE, which do exist for any $\varepsilon > 0$ for all the properties we consider.

Definition 11 (Subgame-perfect ε-NE) For CSG G and $\varepsilon > 0$, a strategy profile $\sigma^\star$ is a *subgame-perfect ε-Nash equilibrium* for objectives $\langle X_i \rangle_{i \in N}$ if and only if $\mathbb{E}^{\sigma^\star}_{\mathsf{G},s}(X_i) \geqslant \sup_{\sigma_i \in \Sigma_i} \mathbb{E}^{\sigma^\star_{-i}[\sigma_i]}_{\mathsf{G},s}(X_i) - \varepsilon$ for all $i \in N$ and $s \in S$.

Example 4. In [14] a non-probabilistic concurrent game is used to model medium access control. Two users with limited energy share a wireless channel and choose to transmit (t_i) or wait (w_i) and, if both transmit, the transmissions fail due to interference. We extend this to a CSG by assuming that transmissions succeed with probability q if both transmit. Figure 2 presents a CSG model of the protocol where each user has enough energy for one transmission. The states are labelled with the status of each user, where the first value represents if the user i has transmitted or not transmitted their message (tr_i and nt_i respectively) and the second if there is sufficient energy to transmit or not (1 and 0 respectively).

If the objectives are to maximise the probability of a successful transmission, there are two subgame-perfect SWNE profiles, one when user 1 waits for user 2 to transmit before transmitting and another when user 2 waits for user 1 to transmit before transmitting. Under both profiles, both users successfully transmit with probability 1. If the objectives are to maximise the probability of being one of the first to transmit, then there is only one SWNE profile corresponding to both users immediately trying to transmit. In this case the probability of each user successfully transmitting is q. ■

3 Property Specification: Extending the Logic rPATL

In order to formalise properties of CSGs, we propose an extension of the logic rPATL, previously defined for zero-sum properties of TSGs [23]. In particular, we add operators to specify nonzero-sum properties, using (social welfare or social cost) Nash equilibria, and provide a semantics for this extended logic on CSGs.

Definition 12 (Extended rPATL syntax) The syntax of our extended version of rPATL is given by the grammar:

$$
\begin{aligned}
\phi &:= \mathtt{true} \mid \mathtt{a} \mid \neg\phi \mid \phi\wedge\phi \mid \langle\!\langle C\rangle\!\rangle \mathtt{P}_{\sim q}[\,\psi\,] \mid \langle\!\langle C\rangle\!\rangle \mathtt{R}^{r}_{\sim x}[\,\rho\,] \mid \langle\!\langle C{:}C'\rangle\!\rangle_{\mathrm{opt}\sim x}(\theta)\\
\psi &:= \mathtt{X}\,\phi \mid \phi\ \mathtt{U}^{\leqslant k}\ \phi \mid \phi\ \mathtt{U}\ \phi\\
\rho &:= \mathtt{I}^{=k} \mid \mathtt{C}^{\leqslant k} \mid \mathtt{F}\ \phi\\
\theta &:= \mathtt{P}[\,\psi\,]{+}\mathtt{P}[\,\psi\,] \mid \mathtt{R}^{r}[\,\rho\,]{+}\mathtt{R}^{r}[\,\rho\,]
\end{aligned}
$$

where $\mathtt{a}$ is an atomic proposition, C and C' are coalitions of players such that $C' = N\backslash C$, opt $\in \{\min,\max\}$, $\sim\in\{<,\leqslant,\geqslant,>\}$, $q\in\mathbb{Q}\cap[0,1]$, $x\in\mathbb{Q}$, r is a reward structure and $k\in\mathbb{N}$.

rPATL is a branching-time temporal logic for stochastic games, which combines the probabilistic operator $\mathtt{P}$ of PCTL [38], PRISM's reward operator $\mathtt{R}$ [44], and the coalition operator $\langle\!\langle C\rangle\!\rangle$ of ATL [5]. The syntax distinguishes between state (ϕ), path (ψ) and reward (ρ) formulae. State formulae are evaluated over states of a CSG, while path and reward formulae are both evaluated over paths.

The core operators from the existing version of rPATL [23] are $\langle\!\langle C\rangle\!\rangle \mathtt{P}_{\sim q}[\,\psi\,]$ and $\langle\!\langle C\rangle\!\rangle \mathtt{R}^{r}_{\sim x}[\,\rho\,]$. A state satisfies a formula $\langle\!\langle C\rangle\!\rangle \mathtt{P}_{\sim q}[\,\psi\,]$ if the *coalition* of players C can ensure that the probability of the path formula ψ being satisfied is $\sim q$, regardless of the actions of the other players ($N\backslash C$) in the game. A state satisfies a formula $\langle\!\langle C\rangle\!\rangle \mathtt{R}^{r}_{\sim x}[\,\rho\,]$ if the players in C can ensure that the expected value of the reward formula ρ for reward structure r is $\sim x$, whatever the other players do. Such properties are inherently *zero-sum* in nature as one coalition tries to maximise an objective (e.g., the probability of ψ) and the other tries to minimise it; hence, we call these *zero-sum formulae.*

The most significant extension we make to the rPATL logic is the addition of *nonzero-sum formulae.* These take the form $\langle\!\langle C{:}C'\rangle\!\rangle_{\mathrm{opt}\sim x}(\theta)$, where C and C' are two coalitions that represent a partition of the set of players N, and θ is the sum of either two probabilistic or two reward objectives. Their meaning is as follows:

- $\langle\!\langle C{:}C'\rangle\!\rangle_{\max\sim x}(\theta)$ is satisfied if there exists a subgame-perfect SWNE profile between coalitions C and C' under which the *sum* of the objectives of C and C' in θ is $\sim x$;

- $\langle\langle C{:}C'\rangle\rangle_{\min\sim x}(\theta)$ is satisfied if there exists a subgame-perfect SCNE profile between coalitions C and C' under which the *sum* of the objectives of C and C' in θ is $\sim x$.

Like the existing zero-sum formulae, the new nonzero-sum formulae still split the players into just two coalitions, C and $C' = N \backslash C$. This means that the model checking algorithm (see Section 4) reduces to finding equilibria in two-player CSGs, which is more tractable than for larger numbers of players. Technically, therefore, we could remove the second coalition C' from the syntax. However, we retain it for clarity about which coalition corresponds to each of the two objectives, and to allow a later extension to more than two coalitions [47].

Both types of formula, zero-sum and nonzero-sum, are composed of *path* (ψ) and *reward* (ρ) formulae, used in probabilistic and reward objectives included within $\mathtt{P}$ and $\mathtt{R}$ operators, respectively. For path formulae, we follow the existing rPATL syntax from [23] and allow *next* ($\mathtt{X}\,\phi$), *bounded until* ($\phi\ \mathtt{U}^{\leqslant k}\ \phi$) and *unbounded until* ($\phi\ \mathtt{U}\ \phi$). We also allow the usual equivalences such as $\mathtt{F}\ \phi \equiv \mathtt{true}\ \mathtt{U}\ \phi$ (i.e., *probabilistic reachability*) and $\mathtt{F}^{\leqslant k}\ \phi \equiv \mathtt{true}\ \mathtt{U}^{\leqslant k}\ \phi$ (i.e., *bounded probabilistic reachability*).

For reward formulae, we introduce some differences with respect to [23]. We allow instantaneous (state) reward at the kth step (*instantaneous reward* $\mathtt{I}^{=k}$), reward accumulated over k steps (*bounded cumulative reward* $\mathtt{C}^{\leqslant k}$), and reward accumulated until a formula ϕ is satisfied (*expected reachability* $\mathtt{F}\ \phi$). The first two, adapted from the property specification language of PRISM [44], were not previously included in rPATL, but proved to be useful for the case studies we present later in Section 7.2. For the third ($\mathtt{F}\ \phi$), [23] defines several variants, which differ in the treatment of paths that never reach a state satisfying ϕ. We restrict our attention to the most commonly used one, which is the default in PRISM, where paths that never satisfy ϕ have infinite reward. In the case of zero-sum formulae, adding the additional variants is straightforward based on the algorithm of [23]. On the other hand, for nonzero-sum formulae, currently no algorithms exist for these variants.

As for other probabilistic temporal logics, it is useful to consider *numerical* queries, which represent the value of an objective, rather than checking whether it is above or below some threshold. In the case of zero-sum formulae, these take the form $\langle\langle C\rangle\rangle\mathtt{P}_{\min=?}[\,\psi\,]$, $\langle\langle C\rangle\rangle\mathtt{P}_{\max=?}[\,\psi\,]$, $\langle\langle C\rangle\rangle\mathtt{R}^{r}_{\min=?}[\,\rho\,]$ and $\langle\langle C\rangle\rangle\mathtt{R}^{r}_{\max=?}[\,\rho\,]$. For nonzero-sum formulae, numerical queries are of the form $\langle\langle C{:}C'\rangle\rangle_{\max=?}[\theta]$ and $\langle\langle C{:}C'\rangle\rangle_{\min=?}[\theta]$ which return the SWNE and SCNE values, respectively.

Example 5. Consider a scenario in which two robots (rbt_1 and rbt_2) move concurrently over a square grid of cells, where each is trying to reach their individual goal location. Each step of the robot involves transitioning to an adjacent cell, possibly stochastically. Examples of *zero-sum* formulae, where $\mathsf{crash}, \mathsf{goal}_1, \mathsf{goal}_2$ denote the obvious atomic propositions labelling states, include:

- $\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\neg\mathsf{crash}\ \mathtt{U}^{\leqslant 10}\ \mathsf{goal}_1\,]$ asks what is the maximum probability with which the first robot can ensure that it reaches its goal location within 10 steps and without crashing, no matter how the second robot behaves;
- $\langle\langle rbt_2\rangle\rangle\mathtt{R}^{r_{\mathsf{crash}}}_{\leqslant 1.5}[\,\mathtt{F}\ \mathsf{goal}_2\,]$ states that, no matter the behaviour of the first robot, the second robot can ensure the expected number of times it crashes before reaching its goal is less than or equal to 1.5 (r_{crash} is a reward structure that assigns 1 to states labelled crash and 0 to all other states).

Examples of *nonzero-sum* formulae include:

- $\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max\geqslant 2}(\mathtt{P}[\,\mathtt{F}\ \mathsf{goal}_1\,]{+}\mathtt{P}[\,\neg\mathsf{crash}\ \mathtt{U}^{\leqslant 10}\mathsf{goal}_2\,])$ states the robots can collaborate so that both reach their goal with probability 1, with the additional condition that the second has to reach its goal within 10 steps without crashing;
- $\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\min=?}(\mathtt{R}^{r_{steps}}[\,\mathtt{F}\ \mathsf{goal}_1\,]{+}\mathtt{R}^{r_{steps}}[\,\mathtt{F}\ \mathsf{goal}_2\,])$ asks what is the sum of expected reachability values when the robots collaborate and each wants to minimise their expected steps to reach their goal (r_{steps} is a reward structure that assigns 1 to all state and action tuple pairs).

Examples of more complex *nested* formulae for this scenario include the following, where r_{steps} is as above:

- $\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ \langle\langle rbt_2\rangle\rangle\mathtt{R}^{r_{steps}}_{\geqslant 10}[\,\mathtt{F}\,\mathsf{goal}_2\,]\,]$ asks what is the maximum probability with which the first robot can get to a state where the expected time for the second robot to reach their goal is at least 10 steps;
- $\langle\langle rbt_1, rbt_2\rangle\rangle\mathtt{P}_{\geqslant 0.75}[\,\mathtt{F}\ \langle\langle rbt_1{:}rbt_2\rangle\rangle_{\min\leqslant 5}(\mathtt{R}^{r_{steps}}[\,\mathtt{F}\,\mathsf{goal}_1\,]{+}\mathtt{R}^{r_{steps}}[\,\mathtt{F}\,\mathsf{goal}_2\,])\,]$ states the robots can collaborate to reach, with probability at least 0.75, a state where the sum of the expected time for the robots to reach their goals is at most 5.

■

Before giving the semantics of the logic, we define *coalition games* which, for a CSG G and coalition (set of players) $C \subseteq N$, reduce G to a two-player CSG G^C, with one player representing C and the other $N\backslash C$. Without loss of generality we assume the coalition of players is of the form $C = \{1,\dots,n'\}$.

Definition 13 (Coalition game) For CSG $\mathsf{G} = (N, S, \bar{s}, A, \Delta, \delta, AP, L)$ and coalition $C = \{1,\dots,n'\} \subseteq N$, the *coalition game* $\mathsf{G}^C = (\{1,2\}, S, \bar{s}, A^C, \Delta^C, \delta^C, AP, L)$ is a two-player CSG where:

- $A^C = (A^C_1 \cup \{\bot\}){\times}(A^C_2 \cup \{\bot\})$;
- $A^C_1 = (A_1 \cup \{\bot\}){\times}\cdots{\times}(A_{n'} \cup \{\bot\}) \setminus \{(\bot,\dots,\bot)\}$;
- $A^C_2 = (A_{n'+1} \cup \{\bot\}){\times}\cdots{\times}(A_n \cup \{\bot\}) \setminus \{(\bot,\dots,\bot)\}$;
- $a^C_1 = (a_1,\dots,a_m) \in \Delta^C(s)$ if and only if either $\Delta(s) \cap A_j = \varnothing$ and $a_j = \bot$ or $a_j \in \Delta(s)$ for all $1 \leqslant j \leqslant m$ and $a^C_2 = (a_{m+1},\dots,a_n) \in \Delta^C(s)$ if and only if either $\Delta(s) \cap A_j = \varnothing$ and $a_j = \bot$ or $a_j \in \Delta(s)$ for all $m+1 \leqslant j \leqslant n$ for $s \in S$;
- for any $s \in S$, $a^C_1 \in A^C_1$ and $a^C_2 \in A^C_2$ we have $\delta^C(s,(a^C_1,a^C_2)) = \delta(s,(a_1,a_2))$ where $a_i = (\bot,\dots,\bot)$ if $a^C_i = \bot$ and $a_i = a^C_i$ otherwise for $1 \leqslant i \leqslant 2$.

Furthermore, for a reward structure $r = (r_A, r_S)$ of G, by abuse of notation we also use r for the corresponding reward structure $r = (r^C_A, r^C_S)$ of G^C where:

- for any $s \in S$, $a^C_1 \in A^C_1$ and $a^C_2 \in A^C_2$ we have $r^C_A(s,(a^C_1,a^C_2)) = r_A(s,(a_1,a_2))$ where $a_i = (\bot,\dots,\bot)$ if $a^C_i = \bot$ and $a_i = a^C_i$ otherwise for $1 \leqslant i \leqslant 2$;
- for any $s \in S$ we have $r^C_S(s) = r_S(s)$.

Our logic includes both *finite-horizon* ($\mathtt{X}$, $\mathtt{U}^{\leqslant k}$, $\mathtt{I}^{=k}$, $\mathtt{C}^{\leqslant k}$) and *infinite-horizon* ($\mathtt{U}$, $\mathtt{F}$) temporal operators. For the latter, the existence of SWNE or SCNE profiles is an open problem [11], but we can check for ε-SWNE or ε-SCNE profiles for any ε. Hence, we define the semantics of the logic in the context of a particular ε.

Definition 14 (Extended rPATL semantics) For a CSG G, $\varepsilon > 0$ and a formula ϕ in our rPATL extension, we define the satisfaction relation $\models$ inductively over the structure of ϕ. The propositional logic fragment ($\texttt{true}$, a, $\neg$, $\wedge$) is defined in the usual way. For a *zero-sum* formula and state $s \in S$ of CSG G, we have:

$$\begin{aligned} s \models \langle\langle C \rangle\rangle \mathtt{P}_{\sim q}[\,\psi\,] &\Leftrightarrow \exists \sigma_1 \in \Sigma^1 . \forall \sigma_2 \in \Sigma^2 . \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G}^C,s}(X^\psi) \sim q \\ s \models \langle\langle C \rangle\rangle \mathtt{R}^r_{\sim x}[\,\rho\,] &\Leftrightarrow \exists \sigma_1 \in \Sigma^1 . \forall \sigma_2 \in \Sigma^2 . \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G}^C,s}(X^{r,\rho}) \sim x \end{aligned}$$

For a *nonzero-sum* formula and state $s \in S$ of CSG G, we have:

$$s \models \langle\langle C{:}C' \rangle\rangle_{\mathrm{opt}\sim x}(\theta) \Leftrightarrow \exists \sigma_1^\star \in \Sigma^1, \sigma_2^\star \in \Sigma^2 . \left(\mathbb{E}^{\sigma_1^\star,\sigma_2^\star}_{\mathsf{G}^C,s}(X_1^\theta) + \mathbb{E}^{\sigma_1^\star,\sigma_2^\star}_{\mathsf{G}^C,s}(X_2^\theta) \right) \sim x$$

where $(\sigma_1^\star, \sigma_2^\star)$ is a subgame-perfect ε-SWNE profile if opt $=$ max, or a subgame-perfect ε-SCNE profile if opt $=$ min, for the objectives (X_1^θ, X_2^θ) in G^C. For an objective X^ψ, $X^{r,\rho}$ or X_i^θ ($1 \leqslant i \leqslant 2$), and path $\pi \in \mathit{IPaths}_{\mathsf{G}^C,s}$:

$$\begin{aligned} X^\psi(\pi) &= 1 \text{ if } \pi \models \psi \text{ and } 0 \text{ otherwise} \\ X^{r,\rho}(\pi) &= \mathit{rew}(r,\rho)(\pi) \\ X_i^{\mathtt{P}[\,\psi^1\,]+\mathtt{P}[\,\psi^2\,]}(\pi) &= 1 \text{ if } \pi \models \psi^i \text{ and } 0 \text{ otherwise} \\ X_i^{\mathtt{R}^{r_1}[\,\rho^1\,]+\mathtt{R}^{r_2}[\,\rho^2\,]}(\pi) &= \mathit{rew}(r_i,\rho^i)(\pi) \end{aligned}$$

For a temporal formula and path $\pi \in \mathit{IPaths}_{\mathsf{G}^C,s}$:

$$\begin{aligned} \pi \models \mathtt{X}\,\phi &\Leftrightarrow \pi(1) \models \phi \\ \pi \models \phi_1 \ \mathtt{U}^{\leqslant k}\ \phi_2 &\Leftrightarrow \exists i \leqslant k . \, (\pi(i) \models \phi_2 \wedge \forall j < i . \, \pi(j) \models \phi_1) \\ \pi \models \phi_1 \ \mathtt{U}\ \phi_2 &\Leftrightarrow \exists i \in \mathbb{N} . \, (\pi(i) \models \phi_2 \wedge \forall j < i . \, \pi(j) \models \phi_1) \end{aligned}$$

For a reward structure r, reward formula and path $\pi \in \mathit{IPaths}_{\mathsf{G}^C,s}$:

$$\begin{aligned} \mathit{rew}(r,\mathtt{I}^{=k})(\pi) &= r_S(\pi(k)) \\ \mathit{rew}(r,\mathtt{C}^{\leqslant k})(\pi) &= \textstyle\sum_{i=0}^{k-1} \big(r_A(\pi(i),\pi[i]) + r_S(\pi(i)) \big) \\ \mathit{rew}(r,\mathtt{F}\ \phi)(\pi) &= \begin{cases} \infty & \text{if } \forall j \in \mathbb{N} . \, \pi(j) \not\models \phi \\ \sum_{i=0}^{k_\phi-1} \big(r_A(\pi(i),\pi[i]) + r_S(\pi(i)) \big) & \text{otherwise} \end{cases} \end{aligned}$$

where $k_\phi = \min\{k \mid \pi(k) \models \phi\}$.

Using the notation above, we can also define the numerical queries mentioned previously. For example, for state s we have:

$$\begin{aligned} \langle\langle C \rangle\rangle \mathtt{P}_{\min=?}[\,\psi\,] &\overset{\text{def}}{=} \inf_{\sigma_1 \in \Sigma^1_{\mathsf{G}^C}} \sup_{\sigma_2 \in \Sigma^2_{\mathsf{G}^C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G}^C,s}(X^\psi) \\ \langle\langle C \rangle\rangle \mathtt{P}_{\max=?}[\,\psi\,] &\overset{\text{def}}{=} \sup_{\sigma_1 \in \Sigma^1_{\mathsf{G}^C}} \inf_{\sigma_2 \in \Sigma^2_{\mathsf{G}^C}} \mathbb{E}^{\sigma_1,\sigma_2}_{\mathsf{G}^C,s}(X^\psi) . \end{aligned}$$

As the zero-sum objectives appearing in the logic are either finite-horizon or infinite-horizon and correspond to either probabilistic until or expected reachability formulae, we have that CSGs are *determined* (see Definition 9) with respect to these objectives [55], and therefore values exist. More precisely, for any CSG G, coalition C, state s, path formula ψ, reward structure r and reward formula ρ, the values $\mathit{val}_{\mathsf{G}^C}(s, X^\psi)$ and $\mathit{val}_{\mathsf{G}^C}(s, X^{r,\rho})$ of the game G^C in state s with respect to the objectives X^{ψ} and $X^{r,\rho}$ are well defined. This determinacy result also yields the following equivalences:

$$\langle\langle C\rangle\rangle\mathtt{P}_{\max=?}[\,\psi\,] \;\equiv\; \langle\langle N\backslash C\rangle\rangle\mathtt{P}_{\min=?}[\,\psi\,] \quad\text{and}\quad \langle\langle C\rangle\rangle\mathtt{R}^{r}_{\max=?}[\,\rho\,] \;\equiv\; \langle\langle N\backslash C\rangle\rangle\mathtt{R}^{r}_{\min=?}[\,\rho\,]\,.$$

Also, as for other probabilistic temporal logics, we can represent negated path formulae by inverting the probability threshold, e.g.: $\langle\langle C\rangle\rangle\mathtt{P}_{\geqslant q}[\,\neg\psi\,] \equiv \langle\langle C\rangle\rangle\mathtt{P}_{\leqslant 1-q}[\,\psi\,]$ and $\langle\langle C{:}C'\rangle\rangle_{\max\geqslant q}(\mathtt{P}[\,\psi_1\,]+\mathtt{P}[\,\psi_2\,]) \equiv \langle\langle C{:}C'\rangle\rangle_{\min\leqslant 2-q}(\mathtt{P}[\,\neg\psi_1\,]+\mathtt{P}[\,\neg\psi_2\,])$, notably allowing the 'globally' operator $\mathtt{G}\;\phi \equiv \neg(\mathtt{F}\;\neg\phi)$ to be defined.

4 Model Checking for Extended rPATL against CSGs

We now present model checking algorithms for the extended rPATL logic, introduced in the previous section, on a CSG G. Since rPATL is a branching-time logic, this works by recursively computing the set $Sat(\phi)$ of states satisfying formula ϕ over the structure of ϕ, as is done for rPATL on TSGs [23].

If ϕ is a zero-sum formula of the form $\langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,]$ or $\langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\rho\,]$, this reduces to computing values for a two-player CSG (either G^{C} or $\mathsf{G}^{N\backslash C}$) with respect to X^{ψ} or $X^{r,\rho}$. In particular, for $\sim\,\in\{\geqslant,>\}$ and $s\in S$ we have:

$$\begin{aligned} s \models \langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,] \;&\Leftrightarrow\; val_{\mathsf{G}^{C}}(s,X^{\psi}) \sim q \\ s \models \langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\rho\,] \;&\Leftrightarrow\; val_{\mathsf{G}^{C}}(s,X^{r,\rho}) \sim x\,. \end{aligned}$$

and, since CSGs are determined for the zero-sum properties we consider, for $\sim\,\in\{<,\leqslant\}$ we have:

$$\begin{aligned} s \models \langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,] \;&\Leftrightarrow\; val_{\mathsf{G}^{N\backslash C}}(s,X^{\psi}) \sim q \\ s \models \langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\rho\,] \;&\Leftrightarrow\; val_{\mathsf{G}^{N\backslash C}}(s,X^{r,\rho}) \sim x\,. \end{aligned}$$

Without loss of generality, for such formulae we focus on computing $val_{\mathsf{G}^{C}}(s,X^{\psi})$ and $val_{\mathsf{G}^{C}}(s,X^{r,\rho})$ and, to simplify the presentation, we denote these values by $\mathtt{V}_{\mathsf{G}^{C}}(s,\psi)$ and $\mathtt{V}_{\mathsf{G}^{C}}(s,r,\rho)$ respectively.

If, on the other hand, ϕ is a nonzero-sum formula of the form $\langle\langle C{:}C'\rangle\rangle_{\mathrm{opt}\sim x}(\theta)$ then, from the semantics for $\langle\langle C{:}C'\rangle\rangle_{\mathrm{opt}\sim x}(\theta)$ (see Definition 14), computing $Sat(\phi)$ reduces to the computation of subgame-perfect SWNE or SCNE values for the objectives $(X^{\theta}_1,X^{\theta}_2)$ and a comparison of their sum to the threshold x. Again, to simplify the presentation, will use the notation $\mathtt{V}_{\mathsf{G}^{C}}(s,\theta)$ for the SWNE values of the objectives $(X^{\theta}_1,X^{\theta}_2)$ in state s of G^{C}.

For the remainder of this section, we fix a CSG $\mathsf{G}=(N,S,\bar{S},A,\Delta,\delta,AP,L)$ and coalition C of players and assume that the available actions of players 1 and 2 of the (two-player) CSG G^{C} in a state s are $\{a_1,\dots,a_l\}$ and $\{b_1,\dots,b_m\}$, respectively. We also fix a value $\varepsilon>0$ which, as discussed in Section 3, is needed to define the semantics of our logic, in particular for infinite-horizon objectives where we need to consider ε-SWNE profiles.

Assumptions. Our model checking algorithms require several assumptions on CSGs, depending on the operators that appear in the formula ϕ. These can all be checked using standard graph algorithms [2]. In the diverse set of model checking case studies that we later present in Section 7.2, these assumptions have not limited the practical applicability of our model checking algorithms.

For zero-sum formulae, the only restriction is for infinite-horizon reward properties on CSGs with both positive and negative reward values.

Assumption 1 *For a zero-sum formula of the form* $\langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\mathtt{F}\ \phi\,]$*, from any state* s *where* $r_S(s) < 0$ *or* $r_A(s,a) < 0$ *for some action* a*, under all profiles of* G*, with probability 1 we reach either a state satisfying* ϕ *or a state where all rewards are zero and which cannot be left with probability 1 under all profiles.*

Without this assumption, the values computed during value iteration can oscillate, and therefore fail to converge (see Appendix A). This restriction is not applied in the existing rPATL model checking algorithms for TSGs [23] since that work assumes that all rewards are non-negative.

The remaining two assumptions concern nonzero-sum formulae that contain infinite-horizon objectives. We restrict our attention to a class of CSGs that can be seen as a variant of *stopping games* [24], as used for multi-objective TSGs. Compared to [24], we use a weaker, objective-dependent assumption, which ensures that, under all profiles, with probability 1, eventually the outcome of each player's objective does not change by continuing.

Assumption 2 *For nonzero-sum formulae, if* $\mathtt{P}[\,\phi_1\ \mathtt{U}\ \phi_2\,]$ *is a probabilistic objective, then* $Sat(\neg\phi_1 \vee \phi_2)$ *is reached with probability 1 from all states under all profiles of* G*.*

Assumption 3 *For nonzero-sum formulae, if* $\mathtt{R}^{r}[\,\mathtt{F}\ \phi\,]$ *is a reward objective, then* $Sat(\phi)$ *is reached with probability 1 from all states under all profiles of* G*.*

Like for Assumption 1, without this restriction, value iteration may not converge since values can oscillate (see Appendices B and C). Notice that Assumption 1 is not required for nonzero-sum properties containing negative rewards since Assumption 3 is itself a stronger restriction.

4.1 Model Checking Zero-Sum Properties

In this section, we present algorithms for *zero-sum* properties, i.e., for computing the values $\mathtt{V}_{\mathsf{G}^C}(s,\psi)$ or $\mathtt{V}_{\mathsf{G}^C}(s,r,\rho)$ for path formulae ψ or reward formulae ρ in all states s of G^C. We split the presentation into *finite-horizon properties*, which can be solved exactly using backward induction [72,60], and *infinite-horizon properties*, for which we approximate values using value iteration [70,19]. Both cases require the solution of matrix games, for which we rely on the linear programming approach presented in Section 2.1.1.

4.1.1 Computing the Values of Zero-Sum Finite-Horizon Formulae

Finite-horizon properties are defined over a bounded number of steps: the next or bounded until operators for probabilistic formulae, and the instantaneous or bounded cumulative reward operators. Computation of the values $\mathtt{V}_{\mathsf{G}^C}(s,\psi)$ or $\mathtt{V}_{\mathsf{G}^C}(s,r,\rho)$ for these is done recursively, based on the step bound, using backward induction and solving matrix games in each state at each iteration. The actions of each matrix game correspond to the actions available in that state; the utilities are constructed from the transition probabilities δ^C of the game G^C, the reward structure r (in the case of reward formulae) and the values already computed recursively for successor states.

Next. This is the simplest operator, over just one step, and so in fact requires no recursion, just solution of a matrix game for each state. If $\psi = \mathtt{X}\,\phi$, then for any state s we have that $\mathtt{V}_{\mathsf{G}^C}(s,\psi) = \mathit{val}(\mathsf{Z})$ where $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ is the matrix game with:

$$z_{i,j} = \textstyle\sum_{s' \in \mathit{Sat}(\phi)} \delta^C(s,(a_i,b_j))(s') \,.$$

Bounded Until. If $\psi = \phi_1 \ \mathtt{U}^{\leqslant k} \ \phi_2$, we compute the values for the path formulae $\psi_n = \phi_1 \ \mathtt{U}^{\leqslant n} \ \phi_2$ for $0 \leqslant n \leqslant k$ recursively. For any state s:

$$\mathtt{V}_{\mathsf{G}^C}(s,\psi_n) = \begin{cases} 1 & \text{if } s \in \mathit{Sat}(\phi_2) \\ 0 & \text{else if } s \notin \mathit{Sat}(\phi_1) \\ 0 & \text{else if } n = 0 \\ \mathit{val}(\mathsf{Z}) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z})$ equals the value of the matrix game $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ with:

$$z_{i,j} = \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v_{n-1}^{s'}$$

and $v_{n-1}^{s'} = \mathtt{V}_{\mathsf{G}^C}(s',\psi_{n-1})$ for all $s' \in S$.

Instantaneous Rewards. If $\rho = \mathtt{I}^{=k}$, then for the reward structure r we compute the values for the reward formulae $\rho_n = \mathtt{I}^{=n}$ for $0 \leqslant n \leqslant k$ recursively. For any state s:

$$\mathtt{V}_{\mathsf{G}^C}(s,r,\rho_n) = \begin{cases} r_S(s) & \text{if } n = 0 \\ \mathit{val}(\mathsf{Z}) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z})$ equals the value of the matrix game $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ with:

$$z_{i,j} = \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v_{n-1}^{s'}$$

and $v_{n-1}^{s'} = \mathtt{V}_{\mathsf{G}^C}(s',r,\rho_{n-1})$ for all $s' \in S$.

Bounded Cumulative Rewards. If $\rho = \mathtt{C}^{\leqslant k}$, then for the reward structure r we compute the values for the reward formulae $\rho_n = \mathtt{C}^{\leqslant n}$ for $0 \leqslant n \leqslant k$ recursively. For any state s:

$$\mathtt{V}_{\mathsf{G}^C}(s,r,\rho_n) = \begin{cases} 0 & \text{if } n = 0 \\ \mathit{val}(\mathsf{Z}) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z})$ equals the value of the matrix game $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ with:

$$z_{i,j} = r_A(s,(a_i,b_j)) + r_S(s) + \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v_{n-1}^{s'}$$

and $v_{n-1}^{s'} = \mathtt{V}_{\mathsf{G}^C}(s',r,\rho_{n-1})$ for all $s' \in S$.

4.1.2 Computing the Values of Zero-Sum Infinite-Horizon Formulae

We now discuss how to compute the values $\mathtt{V}_{\mathsf{G}^C}(s,\psi)$ and $\mathtt{V}_{\mathsf{G}^C}(s,r,\rho)$ for infinite-horizon properties, i.e., when the path formula ψ is an until operator, or for the expected reachability variant of the reward formulae ρ. In both cases, we approximate these values using value iteration, adopting a similar recursive computation to the finite-horizon cases above, solving matrix games in each state and at each iteration, which converges in the limit to the desired values.

Following the approach typically taken in probabilistic model checking tools to implement value iteration, we estimate convergence of the iterative computation by checking the maximum relative difference between successive iterations. However, it is known [36] that, even for simpler probabilistic models such as MDPs, this convergence criterion cannot be used to guarantee that the final computed values are accurate to within a specified error bound. Alternative approaches that resolve this by computing lower and upper bounds for each state have been proposed for MDPs (e.g. [36,13]) and extended to both single- and multi-objective solution of TSGs [42,7]; extensions could be investigated for CSGs. Another possibility is to use *policy iteration* (see, e.g., [18]).

Until. If $\psi = \phi_1 \ \mathtt{U}\ \phi_2$, the probability values can be approximated through value iteration using the fact that $\langle \mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}^{\leqslant k}\ \phi_2)\rangle_{k\in\mathbb{N}}$ is a non-decreasing sequence converging to $\mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}\ \phi_2)$. We compute $\mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}^{\leqslant k}\ \phi_2)$ for increasingly large k and estimate convergence as described above, based on the difference between values in successive iterations. However, we can potentially speed up convergence by first precomputing the set of states S^ψ_0 for which the value of the zero-sum objective X^ψ is 0 and the set of states S^ψ_1 for which the value is 1 using standard graph algorithms [2]. We can then apply value iteration to approximate $\mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}\ \phi_2) = \lim_{k\to\infty} \mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}\ \phi_2, k)$ where:

$$\mathtt{V}_{\mathsf{G}^C}(s,\phi_1 \ \mathtt{U}\ \phi_2, n) = \begin{cases} 1 & \text{if } s \in S^\psi_1 \\ 0 & \text{else if } s \in S^\psi_0 \\ 0 & \text{else if } n = 0 \\ \mathit{val}(\mathsf{Z}) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z})$ equals the value of the matrix game $\mathsf{Z} \in \mathbb{Q}^{l\times m}$ with:

$$z_{i,j} = \textstyle\sum_{s'\in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s'}_{n-1}$$

and $v^{s'}_{n-1} = \mathtt{V}_{\mathsf{G}^C}(s',\phi_1 \ \mathtt{U}\ \phi_2, n-1)$ for all $s' \in S$.

Expected Reachability. If $\rho = \mathtt{F}\ \phi$ and the reward structure is r, then we first make all states of G^C satisfying ϕ absorbing, i.e., we remove all outgoing transitions from such states. Second, we find the set of states S^ρ_∞ for which the reward is infinite; as in [23], this involves finding the set of states satisfying the formula $\langle\langle C\rangle\rangle \mathtt{P}_{<1}[\,\mathtt{F}\ \phi\,]$ and we can use the graph algorithms of [2] to find these states. Again following [23], to deal with zero-reward cycles we need to use value iteration to compute a greatest fixed point. This involves first computing upper bounds on the actual values, by changing all zero reward values to some value $\gamma > 0$ to construct the reward structure $r_\gamma = (r_A^\gamma, r_A^\gamma)$ and then applying value iteration to approximate $\mathtt{V}_{\mathsf{G}^C}(s, r_\gamma, \rho) = \lim_{k\to\infty} \mathtt{V}_{\mathsf{G}^C}(s, r_\gamma, \rho_k)$ where:

$$\mathtt{V}_{\mathsf{G}^C}(s, r_\gamma, \rho_n) = \begin{cases} 0 & \text{if } s \in \mathit{Sat}(\phi) \\ \infty & \text{if } s \in S_\infty^\rho \\ \mathit{val}(\mathsf{Z}) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z})$ equals the value of the matrix game $\mathsf{Z} \in \mathbb{Q}^{l\times m}$ with:

$$z_{i,j} = r_A^\gamma(s,(a_i,b_j)) + r_S^\gamma(s) + \textstyle\sum_{s'\in S} \delta^C(s,(a_i,b_j))(s') \cdot v_{n-1}^{s'}$$

and $v_{n-1}^{s'} = \mathtt{V}_{\mathsf{G}^C}(s', r_\gamma, \rho_{n-1})$ for all $s' \in S$. Finally, using these upper bounds as the initial values we again perform value iteration as above, except now using the original reward structure r, i.e., to approximate $\mathtt{V}_{\mathsf{G}^C}(s, r, \rho) = \lim_{k\to\infty} \mathtt{V}_{\mathsf{G}^C}(s, r, \rho_k)$. The choice of γ can influence value iteration computations in opposing ways: increasing γ can speed up convergence when computing over-approximations, while potentially slowing it down when computing the actual values.

4.2 Model Checking Nonzero-Sum Properties

Next, we show how to compute subgame-perfect SWNE and SCNE values for the two objectives corresponding to a *nonzero-sum* formula. As for the zero-sum case, the approach taken depends on whether the formula contains finite-horizon or infinite-horizon objectives. We now have three cases:

1. when both objectives are finite-horizon, we use backward induction [72,60] to compute (precise) subgame-perfect SWNE and SCNE values;
2. when both objectives are infinite-horizon, we use value iteration [70,19] to approximate the values;
3. when there is a mix of the two types of objectives, we convert the problem to two infinite-horizon objectives on an augmented model.

We describe these three cases separately in Sections 4.2.1, 4.2.2 and 4.2.3, respectively, focusing on the computation of SWNE values. Then, in Section 4.2.4, we explain how to adapt this for SCNE values.

In a similar style to the algorithms for zero-sum properties, in all three cases the computation is an iterative process that analyses a two-player game for each state at each step. However, this now requires finding SWNE or SCNE values of a bimatrix game, rather than solving a matrix game as in the zero-sum case. We solve bimatrix games using the approach presented in Section 2.1.2 (see also the more detailed discussion of its implementation in Section 6.2).

Another important aspect of our algorithms is that, for efficiency, if we reach a state where the value of one player's objective cannot change (e.g., the goal of that player is reached or can no longer be reached), then we switch to the simpler problem of solving an MDP to find the optimal value for the other player in that state. This is possible since the only SWNE profile in that state corresponds to maximising the objective of the other player. More precisely:

– the first player (whose objective cannot change) is *indifferent*, since its value will not be affected by the choices of either player;

– the second player cannot do better than the optimal value of its objective in the corresponding MDP where both players collaborate;
– for any NE profile, the value of the first player is fixed and the value of the second is less than or equal to the optimal value of its objective in the MDP.

We use the notation $\mathtt{P}^{\max}_{\mathsf{G},s}(\psi)$ and $\mathtt{R}^{\max}_{\mathsf{G},s}(r,\rho)$ for the maximum probability of satisfying the path formula ψ and the maximum expected reward for the random variable $rew(r,\rho)$, respectively, when the players collaborate in state s. These values can be computed through standard MDP model checking [10,1].

4.2.1 Computing SWNE Values of Finite-Horizon Nonzero-Sum Formulae

As for the zero-sum case, for a *finite-horizon* nonzero-sum formula θ, we compute the SWNE values $\mathtt{V}_{\mathsf{G}^C}(s,\theta)$ for all states s of G^C in a recursive fashion based on the step bound. We now solve bimatrix games at each step, which are defined in a similar manner to the matrix games for zero-sum properties: the actions of each bimatrix game correspond to the actions available in that state and the utilities are constructed from the transition probabilities δ^C of the game G^C, the reward structure (in the case of reward formulae) and the values already computed recursively for successor states.

For any state formula ϕ and state s we let $\eta_\phi(s)$ equal 1 if $s \in \mathit{Sat}(\phi)$ and 0 otherwise. Recall that probability and reward values of the form $\mathtt{P}^{\max}_{\mathsf{G},s}(\psi)$ and $\mathtt{R}^{\max}_{\mathsf{G},s}(r,\rho)$, respectively, are computed through standard MDP verification. Below, we explain the computation for both types of finite-horizon probabilistic objectives (next and bounded until) and reward objectives (instantaneous and bounded cumulative), as well as combinations of each type.

Next. If $\theta = \mathtt{P}[\,\mathtt{X}\,\phi^1\,]{+}\mathtt{P}[\,\mathtt{X}\,\phi^2\,]$, then $\mathtt{V}_{\mathsf{G}^C}(s,\theta)$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1,\mathsf{Z}_2) \in \mathbb{Q}^{l\times m}$ where:

$$\begin{aligned} z^1_{i,j} &= \textstyle\sum_{s'\in \mathit{Sat}(\phi^1)} \delta^C(s,(a_i,b_j))(s') \\ z^2_{i,j} &= \textstyle\sum_{s'\in \mathit{Sat}(\phi^2)} \delta^C(s,(a_i,b_j))(s')\,. \end{aligned}$$

Again, since next is a 1-step property, no recursion is required.

Bounded Until. If $\theta = \mathtt{P}[\,\phi^1_1\ \mathtt{U}^{\leqslant k_1}\ \phi^1_2\,] + \mathtt{P}[\,\phi^2_1\ \mathtt{U}^{\leqslant k_2}\ \phi^2_2\,]$, we compute SWNE values for the objectives for the nonzero-sum formulae $\theta_{n+n_1,n+n_2} = \mathtt{P}[\,\phi^1_1\ \mathtt{U}^{\leqslant n+n_1}\ \phi^1_2\,] + \mathtt{P}[\,\phi^2_1\ \mathtt{U}^{\leqslant n+n_2}\ \phi^2_2\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \min\{k_1,k_2\}$, $n_1 = k_1{-}k$ and $n_2 = k_2{-}k$. In this case, there are three situations in which the value of the objective of one of the players cannot change, and hence we can switch to MDP verification. The first is when the step bound is zero for only one of the corresponding objectives, the second is when a state satisfying ϕ^i_2 is reached by only one player i (and therefore the objective is satisfied by that state) and the third is when a state satisfying $\neg\phi^i_1 \wedge \neg\phi^i_2$ is reached by only one player i (and therefore the objective is not satisfied by that state). For any state s, if $n = 0$, then:

$$\mathtt{V}_{\mathsf{G}^C}(s,\theta_{n_1,n_2}) = \begin{cases} (\eta_{\phi^1_2}(s), \eta_{\phi^2_2}(s)) & \text{if } n_1 = n_2 = 0 \\ (\eta_{\phi^1_2}(s), \mathtt{P}^{\max}_{\mathsf{G},s}(\phi^2_1\ \mathtt{U}^{\leqslant n_2}\ \phi^2_2)) & \text{else if } n_1 = 0 \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi^1_1\ \mathtt{U}^{\leqslant n_1}\ \phi^1_2), \eta_{\phi^2_2}(s)) & \text{otherwise.} \end{cases}$$

On the other hand, if $n > 0$, then:

$$\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n+n_1,n+n_2}) = \begin{cases} (1,1) & \text{if } s \in \mathit{Sat}(\phi_2^1) \cap \mathit{Sat}(\phi_2^2) \\ (1, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}^{\leqslant n+n_2} \ \phi_2^2)) & \text{else if } s \in \mathit{Sat}(\phi_2^1) \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}^{\leqslant n+n_1} \ \phi_2^1), 1) & \text{else if } s \in \mathit{Sat}(\phi_2^2) \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}^{\leqslant n+n_1} \ \phi_2^1), 0) & \text{else if } s \in \mathit{Sat}(\phi_1^1) \setminus \mathit{Sat}(\phi_1^2) \\ (0, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}^{\leqslant n+n_2} \ \phi_2^2)) & \text{else if } s \in \mathit{Sat}(\phi_1^2) \setminus \mathit{Sat}(\phi_1^1) \\ (0,0) & \text{else if } s \not\in \mathit{Sat}(\phi_1^1) \cap \mathit{Sat}(\phi_1^2) \\ \mathit{val}(\mathsf{Z}_1, \mathsf{Z}_2) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z}_1, \mathsf{Z}_2)$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$:

$$\begin{aligned} z^1_{i,j} &= \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',1}_{(n-1)+n_1} \\ z^2_{i,j} &= \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',2}_{(n-1)+n_2} \end{aligned}$$

and $(v^{s',1}_{(n-1)+n_1}, v^{s',2}_{(n-1)+n_2}) = \mathtt{V}_{\mathsf{G}^C}(s', \theta_{(n-1)+n_1,(n-1)+n_2})$ for all $s' \in S$.

Next and Bounded Until. If $\theta = \mathtt{P}[\,\mathtt{X}\,\phi^1\,] + \mathtt{P}[\,\phi_1^2 \ \mathtt{U}^{\leqslant k_2} \ \phi_2^2\,]$, then $\mathtt{V}_{\mathsf{G}^C}(s,\theta)$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$ where:

$$z^1_{i,j} = \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot \eta_{\phi^1}(s')$$

$$z^2_{i,j} = \begin{cases} 1 & \text{if } s \in \mathit{Sat}(\phi_2^2) \\ 0 & \text{else if } k_2 = 0 \\ \sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}^{\leqslant k_2 - 1} \ \phi_2^2) & \text{else if } \mathit{Sat}(\phi_1^2) \\ 0 & \text{otherwise.} \end{cases}$$

In this case, since the value for objectives corresponding to next formulae cannot change after the first step, we can always switch to MDP verification after this step. The symmetric case is similar.

Instantaneous Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{I}^{=k_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{I}^{=k_2}\,]$, we compute SWNE values of the objectives for the nonzero-sum formulae $\theta_{n+n_1,n+n_2} = \mathtt{R}^{r_1}[\,\mathtt{I}^{=n+n_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{I}^{=n+n_2}\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \min\{k_1, k_2\}$, $n_1 = k_1 - k$ and $n_2 = k_2 - k$. Here, there is only one situation in which the value of the objective of one of the players cannot change: when one of the step bounds equals zero. Hence, this is the only time we switch to MDP verification. For any state s, if $n = 0$, then:

$$\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n_1,n_2}) = \begin{cases} (r^1_S(s), r^2_S(s)) & \text{if } n_1 = n_2 = 0 \\ (r^1_S(s), \mathtt{R}^{\max}_{\mathsf{G},s}(r_2, \mathtt{I}^{=n_2})) & \text{else if } n_1 = 0 \\ (\mathtt{R}^{\max}_{\mathsf{G},s}(r_1, \mathtt{I}^{=n_1}), r^2_S(s)) & \text{otherwise.} \end{cases}$$

On the other hand, if $n > 0$, then $\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n+n_1,n+n_2})$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$ where:

$$\begin{aligned} z^1_{i,j} &= \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',1}_{(n-1)+n_1} \\ z^2_{i,j} &= \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',2}_{(n-1)+n_2} \end{aligned}$$

and $(v^{s',1}_{(n-1)+n_1}, v^{s',2}_{(n-1)+n_2}) = \mathtt{V}_{\mathsf{G}^C}(s', \theta_{(n-1)+n_1,(n-1)+n_2})$ for all $s' \in S$.

Bounded Cumulative Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{C}^{\leqslant k_1}\,]+\mathtt{R}^{r_2}[\,\mathtt{C}^{\leqslant k_2}\,]$, we compute values of the objectives for the formulae $\theta_{n+n_1,n+n_2} = \mathtt{R}^{r_1}[\,\mathtt{C}^{\leqslant n+n_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{C}^{\leqslant n+n_2}\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \min\{k_1, k_2\}$, $n_1 = k_1{-}k$ and $n_2 = k_2{-}k$. As for instantaneous rewards, the only time we can switch to MDP verification is when one of the step bounds equals zero. For state s, if $n = 0$:

$$\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n_1,n_2}) = \begin{cases} (0,0) & \text{if } n_1 = n_2 = 0 \\ (0, \mathtt{R}^{\max}_{\mathsf{G},s}(r_2, \mathtt{C}^{\leqslant n_2})) & \text{else if } n_1 = 0 \\ (\mathtt{R}^{\max}_{\mathsf{G},s}(r_1, \mathtt{C}^{\leqslant n_1}), 0) & \text{otherwise} \end{cases}$$

and if $n > 0$, then $\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n+n_1,n+n_2})$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$:

$$\begin{aligned} z^1_{i,j} &= r^1_S(s) + r^1_A(s,(a_i,b_j)) + \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',1}_{(n-1)+n_1} \\ z^2_{i,j} &= r^2_S(s) + r^2_A(s,(a_i,b_j)) + \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',l}_{(n-1)+n_2} \end{aligned}$$

and $(v^{s',1}_{(n-1)+n_1}, v^{s',2}_{(n-1)+n_2}) = \mathtt{V}_{\mathsf{G}^C}(s', \theta_{(n-1)+n_1,(n-1)+n_2})$ for all $s' \in S$.

Bounded Instantaneous and Cumulative Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{I}^{=k_1}\,]+\mathtt{R}^{r_2}[\,\mathtt{C}^{\leqslant k_2}\,]$, we compute values of the objectives for the formulae $\theta_{n+n_1,n+n_2} = \mathtt{R}^{r_1}[\,\mathtt{I}^{=n+n_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{C}^{\leqslant n+n_2}\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \min\{k_1, k_2\}$, $n_1 = k_1{-}k$ and $n_2 = k_2{-}k$. Again, here we can only switch to MDP verification when one of the step bounds equals zero. For state s, if $n = 0$:

$$\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n_1,n_2}) = \begin{cases} (r^1_S(s), 0) & \text{if } n_1 = n_2 = 0 \\ (r^1_S(s), \mathtt{R}^{\max}_{\mathsf{G},s}(r_2, \mathtt{C}^{\leqslant n_2})) & \text{else if } n_1 = 0 \\ (\mathtt{R}^{\max}_{\mathsf{G},s}(r_1, \mathtt{I}^{= n_1}), 0) & \text{otherwise} \end{cases}$$

and if $n > 0$, then $\mathtt{V}_{\mathsf{G}^C}(s, \theta_{n+n_1,n+n_2})$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$:

$$\begin{aligned} z^1_{i,j} &= \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',1}_{(n-1)+n_1} \\ z^2_{i,j} &= r^2_S(s) + r^2_A(s,(a_i,b_j)) + \textstyle\sum_{s' \in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',l}_{(n-1)+n_2} \end{aligned}$$

and $(v^{s',1}_{(n-1)+n_1}, v^{s',2}_{(n-1)+n_2}) = \mathtt{V}_{\mathsf{G}^C}(s', \theta_{(n-1)+n_1,(n-1)+n_2})$ for all $s' \in S$. The symmetric case follows similarly.

4.2.2 Computing SWNE Values of Infinite-Horizon Nonzero-Sum Formulae

We next show how to compute SWNE values $\mathtt{V}_{\mathsf{G}^C}(s, \theta)$ for *infinite-horizon* nonzero-sum formulae θ in all states s of G^C. As for the zero-sum case, we approximate these using a value iteration approach. Each step of this computation is similar in nature to the algorithms in the previous section, where a bimatrix game is solved for each state, and a reduction to solving an MDP is used after one of the player's objective can no longer change.

A key aspect of the value iteration algorithm is that, while the SWNE (or SCNE) values take the form of a pair, with one value for each player, convergence is defined over the *sum* of the two values. This is because there is not necessarily a unique pair of such values, but the maximum (or minimum) of the sum of NE values *is* uniquely defined. Convergence of value iteration is estimated in the same way as for the zero-sum computation (see Section 4.1.2), by comparing values in successive iterations. As previously, this means that we are not able to guarantee that the computed values are within a particular error bound of the exact values.

Below, we give the algorithms for the cases of two infinite-horizon objectives. The notation used is as in the previous section: for any state formula ϕ and state s we let $\eta_\phi(s)$ equal 1 if $s \in \mathit{Sat}(\phi)$ and 0 otherwise; and values of the form $\mathtt{P}^{\max}_{\mathsf{G},s}(\psi)$ and $\mathtt{R}^{\max}_{\mathsf{G},s}(r,\rho)$ are computed through standard MDP verification.

Until. If $\theta = \mathtt{P}[\,\phi^1_1 \ \mathtt{U}\ \phi^1_2\,]+\mathtt{P}[\,\phi^2_1 \ \mathtt{U}\ \phi^2_2\,]$, values for any state s can be computed through value iteration as the limit $\mathtt{V}_{\mathsf{G}^C}(s,\theta) = \lim_{n\to\infty} \mathtt{V}_{\mathsf{G}^C}(s,\theta,n)$ where:

$$\mathtt{V}_{\mathsf{G}^C}(s,\theta,n) = \begin{cases} (1,1) & \text{if } s \in \mathit{Sat}(\phi^1_2) \cap \mathit{Sat}(\phi^2_2) \\ (1,\mathtt{P}^{\max}_{\mathsf{G},s}(\phi^2_1 \ \mathtt{U}\ \phi^2_2)) & \text{else if } s \in \mathit{Sat}(\phi^1_2) \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi^1_1 \ \mathtt{U}\ \phi^1_2),1) & \text{else if } s \in \mathit{Sat}(\phi^2_2) \\ (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi^1_1 \ \mathtt{U}\ \phi^1_2),0) & \text{else if } s \in \mathit{Sat}(\phi^1_1) \setminus \mathit{Sat}(\phi^2_1) \\ (0,\mathtt{P}^{\max}_{\mathsf{G},s}(\phi^2_1 \ \mathtt{U}\ \phi^2_2)) & \text{else if } s \in \mathit{Sat}(\phi^2_1) \setminus \mathit{Sat}(\phi^1_1) \\ (0,0) & \text{else if } n=0 \text{ or } s \notin \mathit{Sat}(\phi^1_1) \cap \mathit{Sat}(\phi^2_1) \\ \mathit{val}(\mathsf{Z}_1,\mathsf{Z}_2) & \text{otherwise} \end{cases}$$

where $\mathit{val}(\mathsf{Z}_1,\mathsf{Z}_2)$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1,\mathsf{Z}_2) \in \mathbb{Q}^{l\times m}$:

$$\begin{aligned} z^1_{i,j} &= \textstyle\sum_{s'\in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',1}_{n-1} \\ z^2_{i,j} &= \textstyle\sum_{s'\in S} \delta^C(s,(a_i,b_j))(s') \cdot v^{s',2}_{n-1} \end{aligned}$$

and $(v^{s',1}_{n-1}, v^{s',2}_{n-1}) = \mathtt{V}_{\mathsf{G}^C}(s',\theta,n{-}1)$ for all $s' \in S$.

As can be seen, there are two situations in which we switch to MDP verification. These correspond to the two cases where the value of the objective of one of the players cannot change: when a state satisfying ϕ^i_2 is reached for only one player i (and therefore the objective is satisfied by that state) and when a state satisfying $\neg\phi^i_1 \wedge \neg\phi^i_2$ is reached for only one player i (and therefore the objective is not satisfied by that state).

Expected Reachability. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{F}\ \phi^1\,]+\mathtt{R}^{r_2}[\,\mathtt{F}\ \phi^2\,]$, values can be computed through value iteration as the limit $\mathtt{V}_{\mathsf{G}^C}(s,\theta) = \lim_{n\to\infty} \mathtt{V}_{\mathsf{G}^C}(s,\theta,n)$ where:

$$\mathtt{V}_{\mathsf{G}^C}(s,\theta,n) = \begin{cases} (0,0) & \text{if } s \in \mathit{Sat}(\phi^1) \cap \mathit{Sat}(\phi^2) \\ (0,0) & \text{else if } n=0 \\ (0,\mathtt{R}^{\max}_{\mathsf{G},s}(r_2,\mathtt{F}\ \phi^2)) & \text{else if } s \in \mathit{Sat}(\phi^1) \\ (\mathtt{R}^{\max}_{\mathsf{G},s}(r_1,\mathtt{F}\ \phi^1),0) & \text{else if } s \in \mathit{Sat}(\phi^2) \\ \mathit{val}(\mathsf{Z}_1,\mathsf{Z}_2) & \text{otherwise} \end{cases}$$

where $val(\mathsf{Z}_1, \mathsf{Z}_2)$ equals SWNE values of the bimatrix game $(\mathsf{Z}_1, \mathsf{Z}_2) \in \mathbb{Q}^{l \times m}$:

$$\begin{aligned} z^1_{i,j} &= r^1_S(s) + r^1_A(s, (a_i, b_j)) + \textstyle\sum_{s' \in S} \delta^C(s, (a_i, b_j))(s') \cdot v^{s',1}_{n-1} \\ z^2_{i,j} &= r^2_S(s) + r^2_A(s, (a_i, b_j)) + \textstyle\sum_{s' \in S} \delta^C(s, (a_i, b_j))(s') \cdot v^{s',2}_{n-1} \end{aligned}$$

and $(v^{s',1}_{n-1}, v^{s',2}_{n-1}) = \mathtt{V}_{\mathsf{G}^C}(s', \theta, n{-}1)$ for all $s' \in S$.

In this case, the only situation in which the value of the objective of one of the players cannot change is when only one of their goals is reached, i.e., when a state satisfying ϕ^i is reached for only one player i. This is therefore the only time we switch to MDP verification.

4.2.3 Computing SWNE Values of Mixed Nonzero-Sum Formulae

We now present the algorithms for computing SWNE values of nonzero-sum formulae containing a *mixture* of both finite- and infinite-horizon objectives. This is achieved by finding values for a sum of two modified (infinite-horizon) objectives θ' on a modified game G' using the algorithms presented in Section 4.2.2. This approach is based on the standard construction for converting the verification of finite-horizon properties to infinite-horizon properties [68]. We consider the cases when the first objective is finite-horizon and second infinite-horizon; the symmetric cases follow similarly. In each case, the modified game has states of the form (s, n), where s is a state of G^C, $n \in \mathbb{N}$ and the SWNE values $\mathtt{V}_{\mathsf{G}^C}(s, \theta)$ are given by the SWNE values $\mathtt{V}_{\mathsf{G}'}((s, 0), \theta')$. Therefore, since we require the SWNE values for all states of the original game, in the modified game the set of initial states equals $\{(s, 0) \mid s \in S\}$.

Next and Unbounded Until. If $\theta = \mathtt{P}[\,\mathtt{X}\,\phi^1\,] + \mathtt{P}[\,\phi^2_1 \;\mathtt{U}\; \phi^2_2\,]$, then we construct the game $\mathsf{G}' = (\{1,2\}, S', \bar{S}', A^C, \Delta', \delta', \{\mathsf{a}_{\phi^1}, \mathsf{a}_{\phi^2_1}, \mathsf{a}_{\phi^2_2}\}, L')$ where:

- $S' = \{(s, n) \mid s \in S \wedge 0 \leqslant n \leqslant 2\}$ and $\bar{S}' = \{(s, 0) \mid s \in S\}$;
- $\Delta'((s, n)) = \Delta^C(s)$ for all $(s, n) \in S'$;
- for any $(s, n), (s', n') \in S'$ and $a \in A^C$:

$$\delta'((s, n), a)((s', n')) = \begin{cases} \delta^C(s, a)(s') & \text{if } 0 \leqslant n \leqslant 1 \text{ and } n' = n{+}1 \\ \delta^C(s, a)(s') & \text{else if } n = n' = 2 \\ 0 & \text{otherwise;} \end{cases}$$

- for any $(s, n) \in S'$ and $1 \leqslant j \leqslant 2$:
  - $\mathsf{a}_{\phi^1} \in L'((s, n))$ if and only if $s \in Sat(\phi^1)$ and $n = 1$;
  - $\mathsf{a}_{\phi^2_j} \in L'((s, n))$ if and only if $s \in Sat(\phi^2_j)$,

and compute the SWNE values of $\theta' = \mathtt{P}[\,\mathtt{true}\;\mathtt{U}\;\mathsf{a}_{\phi^1}\,] + \mathtt{P}[\,\mathsf{a}_{\phi^2_1}\;\mathtt{U}\;\mathsf{a}_{\phi^2_2}\,]$ for G'.

Bounded and Unbounded Until. If $\theta = \mathtt{P}[\,\phi^1_1 \;\mathtt{U}^{\leqslant k_1}\; \phi^1_2\,] + \mathtt{P}[\,\phi^2_1 \;\mathtt{U}\; \phi^2_2\,]$, then we construct the game $\mathsf{G}' = (\{1,2\}, S', \bar{S}', A^C, \Delta', \delta', \{\mathsf{a}_{\phi^1_1}, \mathsf{a}_{\phi^1_2}, \mathsf{a}_{\phi^2_1}, \mathsf{a}_{\phi^2_2}\}, L')$ where:

- $S' = \{(s, n) \mid s \in S \wedge 0 \leqslant n \leqslant k_1{+}1\}$ and $\bar{S}' = \{(s, 0) \mid s \in S\}$;
- $\Delta'((s, n)) = \Delta^C(s)$ for all $(s, n) \in S'$;

– for any $(s,n),(s',n') \in S'$ and $a \in A^C$:

$$\delta'((s,n),a)((s',n')) = \begin{cases} \delta^C(s,a)(s') & \text{if } 0 \leqslant n \leqslant k_1 \text{ and } n' = n{+}1 \\ \delta^C(s,a)(s') & \text{else if } n = n' = k_1{+}1 \\ 0 & \text{otherwise;} \end{cases}$$

– for any $(s,n) \in S'$ and $1 \leqslant j \leqslant 2$:
  – $\mathsf{a}_{\phi^1_j} \in L'((s,n))$ if and only if $s \in \mathit{Sat}(\phi^1_j)$ and $0 \leqslant n \leqslant k_j$;
  – $\mathsf{a}_{\phi^2_j} \in L'((s,n))$ if and only if $s \in \mathit{Sat}(\phi^2_j)$,

and compute the SWNE values of $\theta' = \mathtt{P}[\,\mathsf{a}_{\phi^1_1}\ \mathtt{U}\ \mathsf{a}_{\phi^1_2}\,]{+}\mathtt{P}[\,\mathsf{a}_{\phi^2_1}\ \mathtt{U}\ \mathsf{a}_{\phi^2_2}\,]$ for G'.

Bounded Instantaneous and Expected Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{I}^{=k_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{F}\ \phi^2\,]$, then we construct the game $\mathsf{G}' = (\{1,2\}, S', \bar{S}', A^C, \Delta', \delta', \{\mathsf{a}_{k_1+1}, \mathsf{a}_{\phi^2}\}, L')$ and reward structures r'_1 and r'_2 where:

– $S' = \{(s,n) \mid s \in S \wedge 0 \leqslant n \leqslant k_1{+}1\}$ and $\bar{S}' = \{(s,0) \mid s \in S\}$;
– $\Delta'((s,n)) = \Delta^C(s)$ for all $(s,n) \in S'$;
– for any $(s,n),(s',n') \in S'$ and $a \in A^C$:

$$\delta'((s,n),a)((s',n')) = \begin{cases} \delta^C(s,a)(s') & \text{if } 0 \leqslant n \leqslant k_1 \text{ and } n' = n{+}1 \\ \delta^C(s,a)(s') & \text{else if } n = n' = k_1{+}1 \\ 0 & \text{otherwise;} \end{cases}$$

– for any $(s,n) \in S'$:
  – $\mathsf{a}_{k_1+1} \in L'((s,n))$ if and only if $n = k_1{+}1$;
  – $\mathsf{a}_{\phi^2} \in L'((s,n))$ if and only if $s \in \mathit{Sat}(\phi^2)$;
– for any $(s,n) \in S'$ and $a \in A^C$:
  – $r^{1'}_A((s,n),a) = 0$ and $r^{1'}_S((s,n)) = r^{1^C}_S(s)$ if $n = k_1$ and $r^{1'}_A((s,n),a) = 0$ and $r^{1'}_S((s,n)) = 0$ otherwise;
  – $r^{2'}_A((s,n),a) = r^{2^C}_A(s)(a)$ and $r^{2'}_S((s,n)) = r^{2^C}_S(s)$,

and compute the SWNE values of $\theta' = \mathtt{R}^{r'_1}[\,\mathtt{F}\ \mathsf{a}_{k_1+1}\,]{+}\mathtt{R}^{r'_2}[\,\mathtt{F}\ \mathsf{a}_{\phi^2}\,]$ for G'.

Bounded Cumulative and Expected Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{C}^{\leqslant k_1}\,] + \mathtt{R}^{r_2}[\,\mathtt{F}\ \phi^2\,]$, then we construct the game $\mathsf{G}' = (\{1,2\}, S', \bar{S}', A^C, \Delta', \delta', \{\mathsf{a}_{k_1}, \mathsf{a}_{\phi^2}\}, L')$ and reward structures r'_1 and r'_2 where:

– $S' = \{(s,n) \mid s \in S \wedge 0 \leqslant n \leqslant k_1\}$ and $\bar{S}' = \{(s,0) \mid s \in S\}$;
– $\Delta'((s,n)) = \Delta^C(s)$ for all $(s,n) \in S'$;
– for any $(s,n),(s',n') \in S'$ and $a \in A^C$:

$$\delta'((s,n),a)((s',n')) = \begin{cases} \delta^C(s,a)(s') & \text{if } 0 \leqslant n \leqslant k_1{-}1 \text{ and } n' = n{+}1 \\ \delta^C(s,a)(s') & \text{else if } n = n' = k_1 \\ 0 & \text{otherwise;} \end{cases}$$

– for any $(s,n) \in S'$:
  – $\mathsf{a}_{k_1} \in L'((s,n))$ if and only if $n = k_1$;
  – $\mathsf{a}_{\phi^2} \in L'((s,n))$ if and only if $s \in \mathit{Sat}(\phi^2)$;
– for any $(s,n) \in S'$ and $a \in A^C$:

- $r_A^{1'}((s,n),a) = r_A^{1^C}(s,a)$ if $0 \leqslant n \leqslant k_1{-}1$ and equals 0 otherwise;
- $r_S^{1'}((s,n)) = r_S^{1^C}(s)$ if $0 \leqslant n \leqslant k_1{-}1$ and equals 0 otherwise;
- $r_A^{2'}((s,n),a) = r_A^{2^C}(s)(a)$ and $r_S^{2'}((s,n)) = r_S^{2^C}(s)$.

and compute the SWNE values of $\theta' = \mathtt{R}^{r'_1}[\,\mathtt{F}\ \mathsf{a}_{k_1}\,]{+}\mathtt{R}^{r'_2}[\,\mathtt{F}\ \mathsf{a}_{\phi^2}\,]$ for G'.

4.2.4 Computing SCNE Values of Nonzero-Sum Formulae

The case for SCNE values follows similarly to the SWNE case using backward induction for finite-horizon properties and value iteration for infinite-horizon properties. There are two differences in the computation. First, when solving MDPs, we find the minimum probability of satisfying path formulae and the minimum expected reward for reward formulae. Second, when solving the bimatrix games constructed during backward induction and value iteration, we find SCNE rather than SWNE values; this is achieved through Lemma 1. More precisely, we negate all the utilities in the game, find the SWNE values of this modified game, then negate these values to obtain SCNE values of the original bimatrix game.

4.3 Strategy Synthesis

In addition to verifying formulae in our extension of rPATL, it is typically also very useful to perform *strategy synthesis*, i.e., to construct a witness to the satisfaction of a property. For each zero-sum formula $\langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,]$ or $\langle\langle C\rangle\rangle\mathtt{R}^r_{\sim x}[\,\rho\,]$ appearing as a sub-formula, this comprises optimal strategies for the players in coalition C (or, equivalently, for player 1 in the coalition game G^C) for the objective X^ψ or $X^{r,\rho}$. For each nonzero-sum formula $\langle\langle C{:}C'\rangle\rangle_{\mathrm{opt}\sim x}(\theta)$ appearing as a sub-formula, this is a subgame-perfect SWNE/SCNE profile for the objectives (X_1^θ, X_2^θ) in the coalition game G^C.

We can perform strategy synthesis by adapting the model checking algorithms described in the previous sections which compute the values of zero-sum objectives and SWNE or SCNE values of nonzero-sum objectives. The type of strategy needed (deterministic or randomised; memoryless or finite-memory) depends on the types of objectives. As discussed previously (in Sections 4.2.2 and 4.1.2), for infinite-horizon objectives our use of value iteration means we cannot guarantee that the values computed are within a particular error bound of the actual values; so, the same will be true of the optimal strategy that we synthesise for such a formula.

Zero-sum properties. For zero-sum formulae, all strategies synthesised are randomised; this is in contrast to checking the equivalent properties against TSGs [23], where deterministic strategies are sufficient. For infinite-horizon objectives, we synthesise memoryless strategies, i.e., a distribution over actions for each state of the game. For finite-horizon objectives, strategies are finite-memory, with a separate distribution required for each state and each time step.

For both types of objectives, we synthesise the strategies whilst computing values using the approach presented in Section 4.1: from the matrix game solved for each state, we extract not just the value of the game, but also an optimal (randomised) strategy for player 1 of G^C in that state. It is also possible to extract the optimal strategy for player 2 in the state by solving the dual LP problem for the matrix game (see Section 2.1.1). For finite-horizon objectives, we retain the choices for all steps; for infinite-horizon objectives, just those from the final step of value iteration are needed.

Nonzero-sum properties. In the case of a nonzero-sum formula, randomisation is again needed for all types of objectives. Similarly to zero-sum formulae above, strategies are generated whilst computing SWNE or SCNE values, using the algorithms presented in Section 4.2. Now, we do this in two distinct ways:

– when solving bimatrix games in each state, we also extract an SWNE/SCNE profile, comprising the distributions over actions for each player of G^C in that state;
– when solving MDPs, we also synthesise an optimal strategy for the MDP [49], which is equivalent to a strategy profile for G^C (in fact, randomisation is not needed for this part).

The final synthesised profile is then constructed by initially following the ones generated when solving bimatrix games, and then switching to the MDP strategies if we reach a state where the value of one player's objective cannot change. This means that all strategies synthesised for nonzero-sum formulae may need memory. As for the zero-sum case, finite-horizon strategies are finite-memory since separate player choices are stored for each state and each time step. But, in addition, for both finite- and infinite-horizon objectives, one bit of memory is required to record that a switch is made to the strategy extracted when solving the MDP.

4.4 Complexity

Due to its overall recursive nature, the complexity of our model checking algorithms are linear in the size of the formula ϕ. In terms of the problems solved for each subformula, finding zero-sum values of a 2-player CSG is PSPACE [20] and finding subgame-perfect NE for reachability objectives of a 2-player CSG is PSPACE-complete [15]. In practice, our algorithms are iterative, so the complexity depends on the number of iterations required, the number of states in the CSG and the problems solved for each state and in each step.

For finite-horizon objectives, the number of iterations is equal to the step-bound in the formula. For infinite-horizon objectives, the number of iterations depends on the convergence criterion used. For zero-sum properties, an exponential lower bound has been shown for the worst-case number of iterations required for a non-trivial approximation [37]. We report on efficiency in practice in Section 7.1.

In the case of zero-sum properties, for each state, at each iteration, we need to solve an LP problem of size $|A|$. Such problems can be solved using the simplex algorithm, which is PSPACE-complete [29], but performs well on average [75]. Alternatively, Karmarkar's algorithm [41] could be used, which is in PTIME.

For nonzero-sum properties, in each state, at each iteration, we need to find all solutions to an LCP problem of size $|A|$. Papadimitriou established the complexity of solving the class of LCPs we encounter to be in PPAD (*polynomial parity argument in a directed graph*) [65] and, to the best of our knowledge, there is still no polynomial algorithm for solving such problems. More closely related to finding all solutions, it has been shown that determining if there exists an equilibrium in a bimatrix game for which each player obtains a utility of a given bound is

NP-complete [32]. Also, it is demonstrated in [8] that bimatrix games may have a number of NE that is exponential with respect to the size of the game, and thus any method that relies on finding all NE in the worst case cannot be expected to perform in a running time that is polynomial with respect to the size of the game.

5 Correctness of the Model Checking Algorithms

The overall (recursive) approach and the reduction to solution of a two-player game is essentially the same as for TSGs [23], and therefore the same correctness arguments apply. In the case of *zero-sum* formulae, the correctness of value iteration for infinite-horizon properties follows from [70] and for finite-horizon properties from Definition 14 and the solution of matrix games (see Section 2). Below, we show the correctness of the model checking algorithms for *nonzero-sum* formulae.

5.1 Nonzero-Sum Formulae

We fix a game G and a nonzero-sum formula $\langle\langle C{:}C'\rangle\rangle_{\mathrm{opt}\sim x}(\theta)$. For the case of *finite-horizon* nonzero-sum formulae, the correctness of the model checking algorithms follows from the fact that we use backward induction [72, 60]. For *infinite-horizon* nonzero-sum formulae, the proof is based on showing that the values computed during value iteration correspond to subgame-perfect SWNE values of finite game trees, and the values of these game trees converge uniformly and are bounded from above by the actual values of G^C.

The fact that we use MDP model checking when the goal of one of the players is reached means that the values computed during value iteration are not finite approximations for the values of G^C. Therefore we must also show that the values computed during value iteration are bounded from below by finite approximations for the values of G^C. We first consider the case when both the objectives in the sum θ are infinite-horizon objectives. Below we assume opt = max and the case when opt = min follow similarly. For any $(v_1, v_2), (v_1', v_2') \in \mathbb{Q}^2$, let $(v_1, v_2) \leqslant (v_1', v_2')$ if and only if $v_1 \leqslant v_1'$ and $v_2 \leqslant v_2'$. The following lemma follows by definition of subgame-perfect SWNE values.

Lemma 2 *Consider any strategy profile σ and state s of G^C and let $(v_1^{\sigma,s}, v_2^{\sigma,s})$ be the corresponding values of the players in s for the objectives $(X^{\theta_1}, X^{\theta_2})$. Considering subgame-perfect SWNE values of the objectives $(X^{\theta_1}, X^{\theta_2})$ in state s, in the case that θ is of the form $\mathtt{P}[\,\phi_1^1 \ \mathtt{U}\ \phi_2^1\,]{+}\mathtt{P}[\,\phi_1^2 \ \mathtt{U}\ \phi_2^2\,]$:*

- *if $s \models \phi_2^1 \wedge \phi_2^2$, then $(1,1)$ are the unique subgame-perfect SWNE values for state s and $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (1,1)$;*
- *if $s \models \phi_2^1 \wedge \phi_1^2 \wedge \neg\phi_2^2$, then $(1, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}\ \phi_2^2))$ are the unique subgame-perfect SWNE values for state s and $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (1, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}\ \phi_2^2))$;*
- *if $s \models \phi_1^1 \wedge \neg\phi_2^1 \wedge \phi_2^2$, then $(\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}\ \phi_2^1), 1)$ are the unique subgame-perfect SWNE values for state s and $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}\ \phi_2^1), 1)$;*
- *if $s \models \phi_2^1 \wedge \neg\phi_1^2 \wedge \neg\phi_2^2$, then $(1,0)$ are the unique subgame-perfect SWNE values for state s and $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (1,0)$;*
- *if $s \models \neg\phi_1^1 \wedge \neg\phi_2^1 \wedge \phi_2^2$, then $(0,1)$ are the unique subgame-perfect SWNE values for state s and $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (0,1)$;*

- *if* $s \models \neg\phi_1^1 \wedge \neg\phi_2^1 \wedge \phi_1^2 \wedge \neg\phi_2^2$, *then* $(0, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}\ \phi_2^2))$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (0, \mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^2 \ \mathtt{U}\ \phi_2^2))$*;*
- *if* $s \models \phi_1^1 \wedge \neg\phi_2^1 \wedge \neg\phi_1^2 \wedge \neg\phi_2^2$, *then* $(\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}\ \phi_2^1), 0)$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (\mathtt{P}^{\max}_{\mathsf{G},s}(\phi_1^1 \ \mathtt{U}\ \phi_2^1), 0)$*;*
- *if* $s \models \neg\phi_1^1 \wedge \neg\phi_2^1 \wedge \neg\phi_1^2 \wedge \neg\phi_2^2$, *then* $(0,0)$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (0,0)$*.*

On the other hand, in the case that θ *is of the form* $\mathtt{R}^{r_1}[\,\mathtt{F}\ \phi^1\,]+\mathtt{R}^{r_2}[\,\mathtt{F}\ \phi^2\,]$ *:*

- *if* $s \models \phi^1 \wedge \phi^2$, *then* $(0,0)$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (0,0)$*;*
- *if* $s \models \phi^1 \wedge \neg\phi^2$, *then* $(0, \mathtt{R}^{\max}_{\mathsf{G},s}(r_2, \mathtt{F}\ \phi^2))$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (0, \mathtt{R}^{\max}_{\mathsf{G},s}(r_2, \mathtt{F}\ \phi^2))$*;*
- *if* $s \models \neg\phi^1 \wedge \phi^2$, *then* $(\mathtt{R}^{\max}_{\mathsf{G},s}(r_1, \mathtt{F}\ \phi^1), 0)$ *are the unique subgame-perfect SWNE values for state* s *and* $(v_1^{\sigma,s}, v_2^{\sigma,s}) \leqslant (\mathtt{R}^{\max}_{\mathsf{G},s}(r_1, \mathtt{F}\ \phi^1), 0)$*.*

Next we require the following objectives of G^C.

Definition 15 For any sum of two probabilistic or reward objectives θ, $1 \leqslant i \leqslant 2$ and $n \in \mathbb{N}$, let $X^\theta_{i,n}$ be the objective where for any path π of G^C :

$$X_{i,n}^{\mathtt{P}[\,\phi_1^1\ \mathtt{U}\ \phi_2^1\,]+\mathtt{P}[\,\phi_1^2\ \mathtt{U}\ \phi_2^2\,]}(\pi) = \begin{cases} 1 & \text{if } \exists k \leqslant n.\, (\pi(k) \models \phi_2^i \wedge \forall j < k.\, \pi(j) \models \phi_1^i) \\ 0 & \text{otherwise} \end{cases}$$

$$X_{i,n}^{\mathtt{R}^{r_1}[\,\mathtt{F}\ \phi^1\,]+\mathtt{R}^{r_2}[\,\mathtt{F}\ \phi^2\,]}(\pi) = \begin{cases} \infty & \text{if } \forall k \in \mathbb{N}.\, \pi(k) \not\models \phi^i \\ \sum_{k=0}^{k_{\phi_i}-1} \big(r_A(\pi(k), \pi[k]) + r_S(\pi(k))\big) & \text{if } k_{\phi^i} \leqslant n{-}1 \\ 0 & \text{otherwise} \end{cases}$$

and $k_{\phi_i} = \min\{k \mid k \in \mathbb{N} \wedge \pi(k) \models \phi^i\}$.

The following lemma demonstrates that, for a fixed strategy profile and state, the values of these objectives are non-decreasing and converge uniformly to the values of θ.

Lemma 3 *For any sum of two probabilistic or reward objectives* θ *and* $\varepsilon > 0$, *there exists* $N \in \mathbb{N}$ *such that, for any* $n \geqslant N$, $s \in S$, $\sigma \in \Sigma^1_{\mathsf{G}^C} \times \Sigma^2_{\mathsf{G}^C}$ *and* $1 \leqslant i \leqslant 2$:

$$0 \ \leqslant\ \mathbb{E}^\sigma_{\mathsf{G}^C,s}(X^\theta_i) - \mathbb{E}^\sigma_{\mathsf{G}^C,s}(X^\theta_{i,n}) \ \leqslant\ \varepsilon\,.$$

Proof Consider any sum of two probabilistic or reward objectives θ, state s and $1 \leqslant i \leqslant 2$. Using Assumption 3 we have that, for subformulae $\mathtt{R}^r[\,\mathtt{F}\ \phi^i\,]$, the set $\mathit{Sat}(\phi^i)$ is reached with probability 1 from all states of G under all profiles, and therefore $\mathbb{E}^\sigma_{\mathsf{G}^C,s}(X^\theta_i)$ is finite. Furthermore, for any $n \geqslant N$, by Definitions 14 and 15 we have that $\mathbb{E}^\sigma_{\mathsf{G}^C,s}(X^\theta_{i,n})$ is the value of state s for the nth iteration of value iteration [19] when computing $\mathbb{E}^\sigma_{\mathsf{G}^C,s}(X^\theta_i)$ in the DTMC obtained from G^C by following the strategy σ, and the sequence is both non-decreasing and converges. The fact that we can choose an N independent of the strategy profile for uniform convergence follows from Assumptions 2 and 3. □

In the proof of correctness we will use the fact that n iterations of value iteration is equivalent to performing backward induction on the following game trees.

Definition 16 For any state s and $n \in \mathbb{N}$, let $\mathsf{G}^C_{n,s}$ be the game tree corresponding to playing G^C for n steps when starting from state s and then terminating.

We can map any strategy profile σ of G^C to a strategy profile of $\mathsf{G}^C_{n,s}$ by only considering the choices of the profile over the first n steps when starting from state s. This mapping is clearly surjective, i.e., we can generate all profiles of $\mathsf{G}^C_{n,s}$, but is not injective. We also need the following objectives corresponding to the values computed during value iteration for the game trees of Definition 16.

Definition 17 For any sum of two probabilistic or reward objectives θ, $s \in S$, $n \in \mathbb{N}$, $1 \leqslant i \leqslant 2$ and $j = i{+}1 \bmod 3$, let Y^θ_i be the objective where, for any path π of $\mathsf{G}^C_{n,s}$:

$$Y_i^{\mathtt{P}[\,\phi_1^1\,\mathtt{U}\,\phi_2^1\,]+\mathtt{P}[\,\phi_1^2\,\mathtt{U}\,\phi_2^2\,]}(\pi) =$$
$$\begin{cases} 1 & \text{if } \exists m \leqslant n.\, (\pi(m) \models \phi^i \wedge \forall k < m.\, \pi(k) \models \phi_1^1 {\wedge} \neg\phi_2^1 {\wedge} \phi_1^2 {\wedge} \neg\phi_2^2) \\ \mathtt{P}^{\max}_{\mathsf{G},\pi(m)}(\phi_1^i \;\mathtt{U}\; \phi_2^i) & \text{else if } \exists m \leqslant n.\, (\pi(m) \models \phi^j \wedge \forall k < m.\, \pi(k) \models \phi_1^1 {\wedge} \neg\phi_2^1 {\wedge} \phi_1^2 {\wedge} \neg\phi_2^2) \\ 0 & \text{otherwise} \end{cases}$$

$$Y_i^{\mathtt{R}^{r_1}[\,\mathtt{F}\,\phi^1\,]+\mathtt{R}^{r_2}[\,\mathtt{F}\,\phi^2\,]}(\pi) =$$
$$\begin{cases} \infty & \text{if } \forall k \leqslant n.\, \pi(k) \not\models \phi^i \\ \sum_{k=0}^{k_{\phi^1 \vee \phi^2}-1} \big(r_A(\pi(k), \pi[k]) + r_S(\pi(k))\big) + r_S^i(\pi(k)) & \text{otherwise} \end{cases}$$

where

$$r_S^i(s') \;=\; \begin{cases} \mathtt{R}^{\max}_{\mathsf{G},s'}(r_i, \mathtt{F}\;\phi^i) & \text{if } s \models \neg\phi^i \wedge \phi^j \\ 0 & \text{otherwise} \end{cases}$$

for $s' \in S$ and $k_{\phi^1 \vee \phi^2} = \min\{k \mid k \leqslant n \wedge \pi(k) \models \phi^1 \vee \phi^2\}$.

Similarly to Lemma 3, the lemma below demonstrates, for a fixed strategy profile and state s of G^C, that the values for the objectives given in Definition 17 when played on the game trees $\mathsf{G}^C_{n,s}$ are non-decreasing and converge uniformly. As with Lemma 3 the result follows from Assumptions 2 and 3.

Lemma 4 *For any sum of two probabilistic or reward objectives θ and $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that for any $m \geqslant n \geqslant N$, $\sigma \in \Sigma^1_{\mathsf{G}^C} \times \Sigma^2_{\mathsf{G}^C}$, $s \in S$ and $1 \leqslant i \leqslant 2$:*

$$0 \;\leqslant\; \mathbb{E}^\sigma_{\mathsf{G}^C_{m,s}}(Y^\theta_i) - \mathbb{E}^\sigma_{\mathsf{G}^C_{n,s}}(Y^\theta_i) \;\leqslant\; \varepsilon\,.$$

We require the following lemma relating the values of the objectives $X^\theta_{i,n}$, Y^θ_i and X^θ_i for $1 \leqslant i \leqslant 2$.

Lemma 5 *For any sum of two probabilistic or reward objectives θ, state s of G^C, strategy profile σ such that when one of the targets of the objectives of θ is reached, the profile then collaborates to maximise the value of the other objective, $n \in \mathbb{N}$ and $1 \leqslant i \leqslant 2$:*

$$\sup\nolimits_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_{i,n}) \;\leqslant\; \sup\nolimits_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) \;\leqslant\; \sup\nolimits_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_i)\,.$$

Proof Consider any strategy profile σ, $n \in \mathbb{N}$ and $1 \leqslant i \leqslant 2$. By Definitions 15 and 17 it follows that:

$$\mathbb{E}^{\sigma}_{\mathsf{G}^C,s}(X^{\theta}_{i,n}) \;\leqslant\; \mathbb{E}^{\sigma}_{\mathsf{G}^C_{n,s}}(Y^{\theta}_i).$$

Furthermore, if we restrict the profile σ such that, when one of the targets of the objectives of θ is reached, the profile then collaborates to maximise the value of the other objective, then by Definitions 17 and 14:

$$\mathbb{E}^{\sigma}_{\mathsf{G}^C_{n,s}}(Y^{\theta}_i) \;\leqslant\; \mathbb{E}^{\sigma}_{\mathsf{G}^C,s}(X^{\theta}_i).$$

Combining these results with Lemma 2, we have:

$$\sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^{\theta}_{i,n}) \;\leqslant\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C_{n,s}}(Y^{\theta}_i) \;\leqslant\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^{\theta}_i)$$

as required. □

We now define the strategy profiles synthesised during value iteration.

Definition 18 For any $n \in \mathbb{N}$ and $s \in S$, let $\sigma^{n,s}$ be the strategy profile generated for the game tree $\mathsf{G}^C_{n,s}$ (when considering value iteration as backward induction) and $\sigma^{n,\star}$ be the synthesised strategy profile for G^C after n iterations.

Before giving the proof of correctness we require the following results.

Lemma 6 *For any state s of G^C, sum of two probabilistic or reward objectives θ and $n \in \mathbb{N}$ we have that $\sigma^{n,s}$ is a subgame-perfect SWNE profile of the CSG $\mathsf{G}^C_{n,s}$ for the objectives $(Y^{\theta_1}, Y^{\theta_2})$.*

Proof The result follows from the fact that value iteration selects SWNE profiles, value iteration corresponds to performing backward induction for the objectives $(Y^{\theta_1}, Y^{\theta_2})$ and backward induction returns a subgame-perfect NE [72,60]. □

The following proposition demonstrates that value iteration converges and depends on Assumptions 2 and 3. Without these assumptions convergence cannot be guaranteed as demonstrated by the counterexamples in Appendix B and Appendix C. Although value iteration converges, unlike value iteration for MDPs or zero-sum games, the generated sequence of values is not necessarily non-decreasing.

Proposition 1 *For any sum of two probabilistic or reward objectives θ and state s, the sequence $\langle \mathtt{V}_{\mathsf{G}^C}(s,\theta,n)\rangle_{n\in\mathbb{N}}$ converges.*

Proof For any state s and $n \in \mathbb{N}$ we can consider $\mathsf{G}^C_{n,s}$ as two-player infinite-action NFGs $\mathsf{N}_{n,s}$ where for $1 \leqslant i \leqslant 2$:

- the set of actions of player i equals the set of strategies of player i in G^C;
- for the action pair (σ_1, σ_2), the utility function for player i returns $\mathbb{E}^{\sigma}_{\mathsf{G}^C_{n,s}}(Y^{\theta}_i)$.

The correctness of this construction relies on the mapping of strategy profiles from the game G^C to $\mathsf{G}^C_{n,s}$ being surjective. Using Lemma 4, we have that the sequence $\langle \mathsf{N}_{n,s}\rangle_{n\in N}$ of NFGs converges uniformly, and therefore, since $\mathtt{V}_{\mathsf{G}^C}(s,\theta,n)$ are subgame-perfect SWNE values of $\mathsf{G}^C_{n,s}$ (see Lemma 6), the sequence $\langle \mathtt{V}_{\mathsf{G}^C}(s,\theta,n)\rangle_{n\in\mathbb{N}}$ also converges. □

A similar convergence result to Proposition 1 has been shown for the simpler case of discounted properties in [30].

Lemma 7 *For any $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that for any $s \in S$ and $1 \leqslant i \leqslant 2$:*

$$\left| \mathbb{E}^{\sigma^{n,\star}}_{\mathsf{G}^C,s}(X^\theta_i) - \mathbb{E}^{\sigma^{n,s}}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) \right| \;\leqslant\; \varepsilon\,.$$

Proof Using Lemma 4 and Proposition 1, we can choose N such that the choices of the profile $\sigma^{n,s}$ agree with those of $\sigma^{n,\star}$ for a sufficient number of steps such that the inequality holds. □

Theorem 2 *For a given sum of two probabilistic or reward objectives θ and $\varepsilon > 0$, there exists $N \in \mathbb{N}$ such that for any $n \geqslant N$ the strategy profile $\sigma^{n,\star}$ is a subgame-perfect ε-SWNE profile of G^C and the objectives $(X^{\theta_1}, X^{\theta_2})$.*

Proof Consider any $\varepsilon > 0$. From Lemma 7 there exists $N_1 \in \mathbb{N}$ such that for any $s \in S$ and $n \geqslant N_1$:

$$\left| \mathbb{E}^{\sigma^{n,\star}}_{\mathsf{G}^C,s}(X^\theta_i) - \mathbb{E}^{\sigma^{n,s}}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) \right| \;\leqslant\; \tfrac{\varepsilon}{2}\,. \tag{5}$$

For any $m \in \mathbb{N}$ and $s \in S$, using Lemma 6 we have that $\sigma^{m,s}$ is a NE of $\mathsf{G}^C_{m,s}$, and therefore for any $m \in \mathbb{N}$, $s \in S$ and $1 \leqslant i \leqslant 2$:

$$\mathbb{E}^{\sigma^{m,s}}_{\mathsf{G}^C_{m,s}}(Y^\theta_i) \;\geqslant\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{m,s}}} \mathbb{E}^{\sigma^{m,s}_{-i}[\sigma_i]}_{\mathsf{G}^C_{m,s}}(Y^\theta_i)\,. \tag{6}$$

From Lemma 3 there exists $N_2 \in \mathbb{N}$ such that for any $n \geqslant N_2$, $s \in S$ and $1 \leqslant i \leqslant 2$:

$$\sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma^{n,\star}_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_i) - \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma^{n,\star}_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_{i,n}) \;\leqslant\; \tfrac{\varepsilon}{2}\,. \tag{7}$$

By construction, $\sigma^{n,\star}$ is a profile for which, if one of the targets of the objectives of θ is reached, the profile maximises the value of the objective. We can thus rearrange (7) and apply Lemma 5 to yield for any $n \geqslant N_2$, $s \in S$ and $1 \leqslant i \leqslant 2$:

$$\sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma^{n,s}_{-i}[\sigma_i]}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) \;\geqslant\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma^{n,\star}_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_i) - \tfrac{\varepsilon}{2}\,. \tag{8}$$

Letting $N = \max\{N_1, N_2\}$, for any $n \geqslant N$, $s \in S$ and $1 \leqslant i \leqslant 2$:

$$\begin{aligned}
\mathbb{E}^{\sigma^{n,\star}}_{\mathsf{G}^C,s}(X^\theta_i) &\;\geqslant\; \mathbb{E}^{\sigma^{n,s}}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) - \tfrac{\varepsilon}{2} && \text{by (5) since } n \geqslant N_1 \\
&\;\geqslant\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C_{n,s}}} \mathbb{E}^{\sigma^{n,s}_{-i}[\sigma_i]}_{\mathsf{G}^C_{n,s}}(Y^\theta_i) - \tfrac{\varepsilon}{2} && \text{by (6)} \\
&\;\geqslant\; \left(\sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma^{n,\star}_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_i) - \tfrac{\varepsilon}{2} \right) - \tfrac{\varepsilon}{2} && \text{by (8) since } n \geqslant N_2 \\
&\;=\; \sup_{\sigma_i \in \Sigma_i^{\mathsf{G}^C}} \mathbb{E}^{\sigma^{n,\star}_{-i}[\sigma_i]}_{\mathsf{G}^C,s}(X^\theta_i) - \varepsilon
\end{aligned}$$

and hence, since $\varepsilon > 0$, $s \in S$ and $1 \leqslant i \leqslant 2$ were arbitrary, $\sigma^{n,\star}$ is a subgame-perfect ε-NE. It remains to show that the strategy profile is a subgame-perfect social welfare optimal ε-NE, which follows from the fact that when solving the bimatrix games during value iteration social welfare optimal NE are returned. □

It remains to consider the model checking algorithms for nonzero-sum properties for which the sum of objectives contains both a finite-horizon and an infinite-horizon objective. In this case (see Section 4.2.3), for a given game G^C and sum of objectives θ, the algorithms first build a modified game G' with states $S' \subseteq S \times \mathbb{N}$ and sum of infinite-horizon objectives θ' and then computes SWNE/SCNE values of θ' in G'. The correctness of these algorithms follows by first showing there exists a bijection between the profiles of G^C and G' and then that, for any profile σ of G^C and σ', the corresponding profile of G' under this bijection, we have:

$$\mathbb{E}^{\sigma}_{\mathsf{G}^C,s}(X^{\theta}_i) = \mathbb{E}^{\sigma'}_{\mathsf{G}',(s,0)}(X^{\theta'}_i)$$

for all states s of G^C and $1 \leqslant i \leqslant 2$. This result follows from the fact that in Section 4.2.3 we used a standard construction for converting the verification of finite-horizon properties to infinite-horizon properties.

6 Implementation and Tool Support

We have implemented support for modelling and automated verification of CSGs in PRISM-games 3.0 [48], which previously only handled TSGs and zero-sum objectives [51]. The PRISM-games tool is available from [80] and the files for the case studies, described in the next section, are available from [81].

6.1 Modelling

We extended the PRISM-games modelling language to support specification of CSGs. The language allows multiple parallel components, called modules, operating both asynchronously and synchronously. Each module's state is defined by a number of finite-valued variables, and its behaviour is defined using probabilistic guarded commands of the form $[a]\ g \to u$, where a is an action label, g is a guard (a predicate over the variables of all modules) and u is a probabilistic state update. If the guard is satisfied then the command is enabled, and the module can (probabilistically) update its variables according to u. The language also allows for the specification of cost or reward structures. These are defined in a similar fashion to the guarded commands, taking the form $[a]\ g : v$ (for action rewards) and $g : v$ (for state rewards), where a is an action label, g is a guard and v is a real-valued expression over variables.

For CSGs, we assign modules to players and, in every state of the model, each player can choose between the enabled commands of the corresponding modules (or, if no command is enabled, the player idles). In contrast to the usual behaviour of PRISM, where modules synchronise on common actions, in CSGs action labels are distinct for each player and the players move concurrently. To allow the updates of variables to depend on the choices of other players, we extend the language by allowing commands to be labelled with lists of actions $[a_1, \ldots, a_n]$. Moreover, updates to variables can be dependent on the new values of other variables being updated in the same concurrent transition, provided there are no cyclic dependencies. This ensures that variables of different players are updated according to a joint probability distribution. Another addition is the possibility of specifying "independent" modules, that is, modules not associated with a specific player, which do not feature nondeterminism and update their own variables when synchronising with other players' actions. Reward definitions are also extended to use action lists, similarly to commands, so that an action reward can depend on the choices taken by multiple players. For further details of the new PRISM-games modelling language, we refer the reader to the tool documentation [80].

6.2 Implementation

PRISM-games constructs a CSG from a given model specification and implements the rPATL model checking and strategy synthesis algorithms from Section 4. We extend existing functionality within the tool, such as modelling and property language parsers, the simulator and basic model checking functionality. We build, store and verify CSGs using an extension of PRISM's 'explicit' model checking engine, which is based on sparse matrices and implemented in Java. For strategy synthesis we have included the option to export the generated strategies to a graphical representation using the Dot language [31].

Computing values (and optimal strategies) of matrix games (see Section 2.1.1), as required for zero-sum formulae, is performed using the LPSolve library [54] via linear programming. This library is based on the revised simplex and branch-and-bound methods. Computing SWNE or SCNE values (and SWNE or SCNE strategies) of bimatrix games (see Section 2.1.2) for nonzero-sum formulae is performed via labelled polytopes through a reduction to SMT. Currently, we implement this in both Z3 [26] and Yices [28]. As an optimised precomputation step, when possible we also search for and filter out *dominated strategies*, which speeds up computation and reduces calls to the solver.

Since bimatrix games can have multiple SWNE values, when selecting SWNE values of such games we choose the SWNE values for which the value of player 1 is maximal. In case player 1 is indifferent, i.e., their utility is the same for all pairs, we choose the SWNE values which maximise the value of player 2. If both players are indifferent, an arbitrary pair of SWNE values is selected.

Table 1 presents experimental results for the time to solve bimatrix games using the Yices and Z3 solvers, as the numbers of actions of the individual games vary. The table also shows the number of NE in each game N, as found when determining the SWNE values, and also the number of NE in N^{-}, as found when determining the SCNE values (see Lemma 1). These games were generated using GAMUT (a suite of game generators) [62] and a time-out of 2 hours was used for the experiments. The results show Yices to be the faster implementation and that the difference in solution time grows as the number of actions increases. Therefore, in our experimental results in the next section, all verification runs use the Yices implementation. The results in Table 1 also demonstrate that the solution time for either solver can vary widely and depends on both the number of NE that need to be found and the structure of the game. For example, when solving the dispersion games, the differences in the solution times for SWNE and SCNE seem to correspond to the differences in the number of NE that need to found. On the other hand, there is no such correspondence between the difference in the solution times for the covariant games.

Regarding the complexity of solving bimatrix games, if each player has n actions, then the number of possible assignments to the supports of the strategy pro-

Bimatrix Game	Actions of each player	SWNE values			SCNE values		
		Solution time (s)		Num. of	Solution time (s)		Num. of
		Yices	Z3	NE in N	Yices	Z3	NE in N^-
Covariant games	2	0.04	0.06	1	0.2	1.1	1
	4	0.04	0.1	3	0.04	0.1	3
	8	0.3	2.7	13	0.3	1.8	21
	12	8.3	77.0	15	11.0	130.2	45
	16	764.8	4,238	103	318.6	2,627	109
Dispersion games	2	0.04	0.08	3	0.04	0.08	3
	4	0.05	0.2	51	0.04	0.1	15
	8	1.8	16.4	6,051	0.2	1.1	255
	12	6,368	t/o	523,251	6.0	38.3	4,095
Majority voting games	2	0.03	0.08	5	0.03	0.07	2
	4	0.05	0.1	15	0.04	0.1	13
	8	0.5	1.0	433	0.2	0.7	186
	12	9.9	22.4	3,585	9.9	16.0	3,072
	16	532.5	2,386	61,441	465.9	2,791	49,153
Randomly generated games	2	0.15	1.9	1	0.03	0.08	1
	4	0.04	0.1	3	0.03	0.09	3
	8	0.4	2.3	13	0.3	1.8	7
	12	45.60	422.0	27	68.0	343.6	31
	16	2,370	t/o	81	1,112	t/o	69

Table 1: Finding SWNE/SCNE values in bimatrix games: comparing SMT solvers.

Player 1 strategy		Player 2 strategy		Utilities
prob. a_1	prob. a_2	prob. b_1	prob. b_2	(u_1, u_2)
0.0	1.0	0.0	1.0	(0.0,4.0)
0.0	1.0	1.0	0.0	(1.0,4.0)
0.0	1.0	0.5	0.5	(0.5,4.0)
1.0	0.0	0.0	1.0	(0.0,2.0)
1.0	0.0	1.0	0.0	(1.0,2.0)
1.0	0.0	0.5	0.5	(0.5,2.0)
0.5	0.5	0.0	1.0	(0.0,3.0)
0.5	0.5	1.0	0.0	(1.0,3.0)
0.5	0.5	0.5	0.5	(0.5,3.0)

Table 2: Possible NE strategies and utilities of the bimatrix game of Example 6.

files (i.e., the action tuples that are chosen with nonzero probability) is $(2^n-1)^2$, which therefore grows exponentially with the number of actions, surpassing 4.2 billion when each player has 16 actions. This particularly affects performance in cases where one or both players are *indifferent* with respect to a given support. More precisely, in such cases, if there is an equilibrium including pure strategies over these supports, then there are also equilibria including mixed strategies over these supports as the indifferent player would get the same utility for *any affine combination* of pure strategies.

Example 6. Consider the following bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c c} & \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \end{array} & \begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix} \end{array} \qquad \mathsf{Z}_2 = \begin{array}{c c} & \begin{array}{cc} b_1 & b_2 \end{array} \\ \begin{array}{c} a_1 \\ a_2 \end{array} & \begin{pmatrix} 2 & 2 \\ 4 & 4 \end{pmatrix} \end{array}$$

Since the entries in the rows for the utility matrix for player 1 are the same and the columns are the same for player 2, it is easy to see that both players are indifferent with respect to their actions. As can be seen in Table 2, all $(2^2-1)^2 = 9$ possible support assignments lead to an equilibrium. ■

For the task of computing non-optimal NE values, the large number of supports can be somewhat mitigated by eliminating *weakly dominated* strategies [61]. However, removing such strategies is not a straightforward task when computing SWNE or SCNE values, since it can lead to the elimination of SWNE or SCNE profiles, and hence also SWNE or SCNE values. For example, if we removed the row corresponding to action a_2 or the column corresponding to action b_1 from the matrices in Example 6 above, then we eliminate a SWNE profile. As the number of actions for each player increases, the number of NE profiles also tends to increase and so does the likelihood of indifference. Naturally, the number of actions also affects the number of variables that have to be allocated, and the number and complexity of assertions passed to the SMT solver. As our method is based on the progressive elimination of support assignments that lead to NE, it takes longer to find SWNE and SCNE values as the number of possible supports grows and further constraints are added each time an equilibrium is found.

7 Case Studies and Experimental Results

To demonstrate the applicability and benefits of our techniques, and to evaluate their performance, we now present results from a variety of case studies. Supporting material for these examples (models and properties) is available from [81]. These can be run with PRISM-games 3.0 [48].

7.1 Efficiency and Scalability

We begin by presenting a selection of results illustrating the performance of our implementation. The experiments were run on a 2.10 GHz Intel Xeon with 16GB of JVM memory. In Table 3, we present the model statistics for the examples used: the number of players, states, transitions and model construction times (details of the case studies themselves follow in the next section). Due to improvements in the modelling language and the model building procedure, some of the model statistics differ from those presented in [45,46]. The main reason is that the earlier version of the implementation did not allow for variables of different players to be updated following a joint probability distribution, which made it necessary to introduce intermediate states in order to specify some of the behaviour. Also, some model statistics differ from [45] since models were modified to meet Assumptions 2 and 3 to enable the analysis of nonzero-sum properties.

Tables 4 and 5 present the model checking statistics when analysing zero-sum and nonzero-sum properties, respectively. In both tables, this includes the maximum and average number of actions of each coalition in the matrix/bimatrix games solved at each step of value iteration and the number of iterations performed. In the case of zero-sum properties including reward formulae of the form $\texttt{F}\ \phi$, value iteration is performed twice (see Section 4.1.2), and therefore the number of iterations for each stage are presented (and separated by a semi-colon). For zero-sum properties, the timing statistics are divided into the time for qualitative (column 'Qual.') and quantitative verification, which includes solving matrix games (column 'Quant.'). For nonzero-sum properties we divide the timing statistics into the time for CSG verification, which includes solving bimatrix games (column 'CSG'),

Case study [parameters]	Param. values	Players	States	Transitions	Constr. time (s)
Robot coordination [l]	4	2	226	6,610	0.1
	8	2	3,970	201,650	1.0
	12	2	20,450	1,221,074	3.9
	16	2	65,026	4,198,450	11.8
	24	2	330,626	23,049,650	62.5
Future markets investors [*months*]	6	3	34,247	144,396	1.5
	12	3	257,301	1,259,620	8.3
	24	3	829,125	4,318,372	25.4
	36	3	1,400,949	7,377,124	59.3
	48	3	1,972,773	10,435,876	70.3
Future markets investors [*months*,*pbar*]	3,0.5	2	3,460	1,2012	0.3
	6,0.5	2	29,703	12,4748	2.5
	12,0.5	2	221,713	1,083,024	8.0
	18,0.5	2	467,497	2,396,784	16.2
User-centric networks [td, K]	1,3	7	32,214	121,659	2.1
	1,4	7	104,897	433,764	6.1
	1,5	7	294,625	1,325,100	17.5
	1,6	7	714,849	3,465,558	42.8
Aloha (deadline) [$b_{\max}$,D]	1,8	3	3,519	5,839	0.2
	2,8	3	14,230	28,895	0.5
	3,8	3	72,566	181,438	2.1
	4,8	3	413,035	1,389,128	9.4
	5,8	3	2,237,981	9,561,201	58.4
Aloha [$b_{\max}$]	2	3	5,111	10,100	0.3
	3	3	22,812	56,693	0.8
	4	3	107,799	355,734	2.5
	5	3	556,168	2,401,113	13.4
	6	3	3,334,681	17,834,254	118.8
Intrusion detection system [*rounds*]	25	2	75	483	0.07
	50	2	150	983	0.08
	100	2	300	1,983	0.1
	200	2	600	3,983	0.1
Jamming radio systems [*chans*,*slots*]	4,6	2	531	45,004	0.5
	4,12	2	1,623	174,796	1.4
	6,6	2	1,061	318,392	2.0
	6,12	2	3,245	1,240,376	6.6
Medium access control [$e_{\max}$]	10	3	10,591	135,915	1.7
	15	3	33,886	457,680	5.0
	20	3	78,181	1,083,645	8.2
	25	3	150,226	2,115,060	14.0
Medium access control [$e_{\max}$,$s_{\max}$]	4,2	3	14,723	129,097	2.2
	4,4	3	18,751	147,441	4.3
	6,4	3	122,948	1,233,976	11.0
	6,6	3	138,916	1,315,860	12.5
Power control [$e_{\max}$, $pow_{\max}$]	40,8	2	32,812	260,924	1.7
	80,8	2	193,396	1,469,896	6.5
	40,16	2	34,590	291,766	1.6
	80,16	2	301,250	2,627,278	10.5

Table 3: Model statistics for the CSG case studies.

and the instances of MDP verification (column 'MDP'). In the case of mixed nonzero-sum properties, i.e., properties including both finite and infinite horizon objectives, we must first build a new game (see Section 4.2.3); the statistics for these CSGs (number of players, states and transitions) are presented in Table 6. Finally, Table 7 presents the timing results for three nested properties. Here we give the time required for verifying the inner and outer formula separately, as well as the number of iterations for value iteration at each stage.

Our results demonstrate significant gains in efficiency with respect to those presented for zero-sum properties in [45] and nonzero-sum properties in [46] (for

Case study & property	Param.	Actions	Val.	Verif. time (s)	
[parameters]	values	max/avg	iters	Qual.	Quant.
Robot coordination	4,4	3,3/2.52,2.52	4	0.02	0.04
$\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\neg\mathtt{c}\,\mathtt{U}^{\leqslant k}\mathtt{g}_1\,]$	8,8	3,3/2.52,2.52	8	0.27	1.1
$[l,k]$	12,12	3,3/2.68,2.68	12	1.61	5.7
	16,16	3,3/2.76,2.76	16	5.88	35.4
	24,24	3,3/2.84,2.84	24	46.5	185.2
Robot coordination	4	3,3/2.52,2.52	10; 9	0.02	0.1
$\langle\langle rbt_1\rangle\rangle\mathtt{R}_{\min=?}[\,\mathtt{F}\,\mathtt{g}_1\,]$	8	3,3/2.52,2.52	15; 14	0.35	4.1
$[l]$	12	3,3/2.68,2.68	20; 19	1.99	23.2
	16	3,3/2.76,2.76	24; 24	7.57	96.5
	24	3,3/2.84,2.84	34; 33	56.9	744.7
Future markets investors	6	8,2/1.54,1.21	14; 14	1.3	6.0
$\langle\langle i_1\rangle\rangle\mathtt{R}_{\max=?}[\,\mathtt{F}\ \mathtt{c}_1\,]$	12	8,2/1.68,1.27	26; 26	14.8	91.7
$[months]$	24	8,2/1.73,1.29	50; 50	94.3	616.0
	36	8,2/1.74,1.29	74; 73	240.4	1,613
	48	8,2/1.74,1.29	98; 97	395.4	2,770
User-centric networks	1,3	16,8/2.11,1.91	15; 1	1.4	182.3
$\langle\langle user\rangle\rangle\mathtt{R}_{\min=?}[\,\mathtt{F}\ \mathtt{f}\,]$	1,4	16,8/2.31,1.92	21; 1	4.4	776.2
$[td, K]$	1,5	16,8/2.46,1.94	25; 1	14.4	2,456
	1,6	16,8/2.60,1.96	29; 1	43.4	6,762
Aloha (deadline)	2,8	4,2/1.01,1.00	24	0.5	0.8
$\langle\langle usr_2, usr_3\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\,\mathtt{s}_{1,2}{\wedge}t{\leqslant}D\,]$	3,8	4,2/1.01,1.00	23	1.8	1.8
$[b_{\max}, D]$	4,8	4,2/1.01,1.00	23	6.5	4.5
	5,8	4,2/1.01,1.00	23	35.9	9.4
Aloha	2	4,2/1.01,1.00	58; 47	0.2	2.1
$\langle\langle usr_2, usr_3\rangle\rangle\mathtt{R}_{\min=?}[\,\mathtt{F}\,\mathsf{sent}_{2,3}\,]$	3	4,2/1.01,1.00	71; 57	0.6	6.0
$[b_{\max}]$	4	4,2/1.01,1.00	109; 86	3.7	33.7
	5	4,2/1.01,1.00	193; 150	26.6	317.6
	6	4,2/1.01,1.00	362; 279	314.5	3,836
Intrusion detection system	25	2,2/1.96,1.96	25	n/a	0.1
$\langle\langle policy\rangle\rangle\mathtt{R}_{\min=?}[\,\mathtt{C}^{\leqslant rounds}\,]$	50	2,2/1.98,1.98	50	n/a	0.5
$[rounds]$	100	2,2/1.99,1.99	100	n/a	1.1
	200	2,2/2.00,2.00	200	n/a	3.6
Jamming radio systems	4,6	3,3/2.17,2.17	7	0.03	0.1
$\langle\langle user\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ sent{\geqslant}slots/2\,]$	4,12	3,3/2.49,2.49	13	0.2	0.5
$[chans, slots]$	6,6	4,4/2.76,2.76	7	0.1	0.3
	6,12	4,4/3.24,3.24	13	0.5	2.0
Jamming radio systems	4,6,6	3,3/2.17,2.17	6	n/a	0.2
$\langle\langle user\rangle\rangle\mathtt{R}_{\max=?}[\,\mathtt{I}^{=k}\,]$	4,12,12	3,3/2.49,2.49	12	n/a	1.0
$[chans, slots, k]$	6,6,6	4,4/2.76,2.76	6	n/a	0.5
	6,12,12	4,4/3.24,3.24	12	n/a	2.9

Table 4: Statistics for CSG zero-sum verification instances.

the latter, a direct comparison with the published results is possible since it uses an identical experimental setup). The gains are primarily due to faster SMT solving and reductions in CSG size as a result of modelling improvements, and specifically the removal of intermediate states as discussed above.

The implementation can analyse models with over 3 million states and almost 18 million transitions; all are solved in under 2 hours and most are considerably quicker. The majority of the time is spent solving matrix or bimatrix games, so performance is affected by the number of choices available within each coalition, rather than the number of players, as well as the number of states. For example, larger instances of the Aloha models are verified relatively quickly since the coalitions have only one choice in many states (the average number of choices is 1.00 for both coalitions). However, for models where players have choices in almost all states, only models with up to hundreds of thousands of states for zero-sum properties and tens of thousands of states for nonzero-sum properties can be verified within 2 hours.

Case study & property [parameters]	Param. values	Actions max/avg	Val. iters.	Verif. time (s)	
				MDP	CSG
Robot coordination	4,4	3,3/2.07,2.07	4	0.02	0.1
$\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max=?}(\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}^{\leqslant k}\mathtt{g}_1\,]+\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}^{\leqslant k}\mathtt{g}_2\,])$	8,8	3,3/2.52,2.52	8	0.45	1.0
$[l,k]$	12,12	3,3/2.68,2.68	12	6.25	5.8
	16,16	3,3/2.76,2.76	16	34.7	22.2
	24,24	3,3/2.84,2.84	24	375.2	1,365
Robot coordination	4,4,6	3,3/2.07,2.07	6	0.02	0.1
$\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max=?}(\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}^{k_1}\mathtt{g}_1\,]+\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}^{k_2}\mathtt{g}_2\,])$	8,8,10	3,3/2.52,2.52	10	0.6	1.2
$[l,k_1,k_2]$	12,12,14	3,3/2.68,2.68	12	7.0	9.3
	16,16,18	3,3/2.76,2.76	18	43.1	153.5
Robot coordination	4,8	3,3/2.10,2.04	21	0.16	3.3
$\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max=?}(\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}^{\leqslant k}\mathtt{g}_1\,]+\mathtt{P}[\,\neg\mathtt{c}\,\mathtt{U}\,\mathtt{g}_2\,])$	4,16	3,3/2.12,2.05	21	0.28	12.0
$[l,k]$	8,8	3,3/2.53,2.51	34	5.51	77.2
	8,16	3,3/2.54,2.52	35	17.2	3,513
Robot coordination	4	3,3/2.07,2.07	8	0.05	0.2
$\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\min=?}(\mathtt{R}[\,\mathtt{F}\,\mathtt{g}_1\,]+\mathtt{R}[\,\mathtt{F}\,\mathtt{g}_2\,])$	8	3,3/2.52,2.52	15	0.44	10.3
$[l]$	12	3,3/2.68,2.68	23	2.12	227.7
	16	3,3/2.84,2.84	28	12.1	3,272
Future markets investors	3	2,2/1.15,1.15	6	0.1	0.2
$\langle\langle i_1{:}i_2\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{F}\,\mathtt{c}_1\,]+\mathtt{R}[\,\mathtt{F}\,\mathtt{c}_2\,])$	6	2,2/1.22,1.22	13	0.8	3.1
$[months]$	12	2,2/1.27,1.27	25	9.1	284.7
	18	2,2/1.29,1.29	37	31.6	4,326
Aloha (deadline)	1,8	2,4/1.00,1.01	23	0.1	0.3
$\langle\langle usr_1{:}usr_2,usr_3\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\,\mathtt{s}_1\,]+\mathtt{P}[\,\mathtt{F}\,\mathtt{s}_{2,3}\,])$	2,8	2,4/1.00,1.00	23	0.4	1.2
$[b_{\max},D]$	3,8	2,4/1.00,1.00	22	2.0	3.4
	4,8	2,4/1.00,1.00	22	7.1	18.5
	5,8	2,4/1.00,1.00	22	39.9	103.0
Aloha	2	2,4/1.00,1.01	54	0.2	0.8
$\langle\langle usr_1{:}usr_2,usr_3\rangle\rangle_{\min=?}(\mathtt{R}[\,\mathtt{F}\,\mathtt{s}_1\,]+\mathtt{R}[\,\mathtt{F}\,\mathtt{s}_{2,3}\,])$	3	2,4/1.00,1.00	62	0.8	3.1
$[b_{\max}]$	4	2,4/1.00,1.00	88	4.5	40.1
	5	2,4/1.00,1.00	145	35.3	187.7
	6	2,4/1.00,1.00	256	453.7	2,396
Medium access control	10,25	2,4/1.91,3.63	25	0.0	65.8
$\langle\langle p_1{:}p_2,p_3\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{C}^{\leqslant k}\,]+\mathtt{R}[\,\mathtt{C}^{\leqslant k}\,])$	15,25	2,4/1.94,3.75	25	0.0	193.3
$[e_{\max},k]$	20,25	2,4/1.95,3.81	25	0.0	394.8
	25,25	2,4/1.96,3.85	25	0.0	565.4
Medium access control	10,20,25	2,4/1.91,3.63	25	0.1	45.8
$\langle\langle p_1{:}p_2,p_3\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{C}^{\leqslant k_1}\,]+\mathtt{R}[\,\mathtt{C}^{\leqslant k_2}\,])$	15,20,25	2,4/1.94,3.75	25	0.3	146.0
$[e_{\max},k]$	20,20,25	2,4/1.95,3.81	25	0.5	267.6
	25,20,25	2,4/1.96,3.85	25	0.9	375.7
Medium access control	4,2	2,4/1.70,2.88	10	0.61	96.2
$\langle\langle p_1{:}p_2,p_3\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\,\mathtt{m}_1\,]+\mathtt{P}[\,\mathtt{F}\,\mathtt{m}_{2,3}\,])$	4,4	2,4/1.64,2.70	12	0.61	22.8
$[e_{\max},s_{\max}]$	6,4	2,4/1.77,3.12	17	7.40	1,639
	6,6	2,4/1.74,3.02	18	4.09	93.6
Medium access control	4,4,4	2,4/1.67,2.70	12	1.64	39.1
$\langle\langle p_1{:}p_2,p_3\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}^{\leqslant k}\,\mathtt{m}_1\,]+\mathtt{P}[\,\mathtt{F}\,\mathtt{m}_{2,3}\,])$	4,4,8	2,4/1.68,2.70	12	3.30	106.3
$[e_{\max},s_{\max},k]$	6,4,6	2,4/1.76,3.02	18	23.4	341.9
	6,4,12	2,4/1.74,3.02	18	60.0	961.8
Power control	40,8	2,2/1.91,1.91	20	4.1	11.7
$\langle\langle p_1{:}p_2\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{F}\,e_1{=}0\,]+\mathtt{R}[\,\mathtt{F}\,e_2{=}0\,])$	80,8	2,2/1.88,1.88	40	34.8	130.3
$[e_{\max},pow_{\max}]$	40,16	2,2/1.95,1.95	40	5.4	11.6
	80,16	2,2/1.98,1.98	80	64.4	211.4
Power control	40,4,20	2,2/1.69,1.69	20	13.3	27.5
$\langle\langle p_1{:}p_2\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{F}\,e_1{=}0\,]+\mathtt{R}[\,\mathtt{C}^{\leqslant k}\,])$	80,4,20	2,2/1.69,1.69	20	83.9	134.2
$[e_{\max},pow_{\max},k]$	40,8,20	2,2/1.91,1.91	20	49.1	84.6
	80,8,20	2,2/1.88,1.88	20	498.6	846.8
Power control	80,4,15,20	2,2/1.69,1.69	20	0.2	4.4
$\langle\langle p_1{:}p_2\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{I}^{=k_1}\,]+\mathtt{R}[\,\mathtt{I}^{=k_2}\,])$	80,4,20,20	2,2/1.69,1.69	20	0.0	6.2
$[e_{\max},pow_{\max},k_1,k_2]$	80,8,15,20	2,2/1.88,1.88	20	0.9	17.0
	80,8,15,20	2,2/1.88,1.88	20	0.0	22.8

Table 5: Statistics for CSG nonzero-sum verification instances.

Case study & property [parameters]	Param. values	Players	States	Transitions
Robot coordination	4,8	2	1,923	56,385
$\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max=?}(\mathtt{P}[\,\neg\mathsf{c}\,\mathtt{U}^{\leqslant k}\,\mathsf{g}_1\,]+\mathtt{P}[\,\neg\mathsf{c}\,\mathtt{U}\,\mathsf{g}_2\,])$	4,16	2	3,491	104,545
$[l,k]$	8,8	2	36,773	1,860,691
	8,16	2	67,525	3,443,443
Medium access control	4,4,4	3	89,405	718,119
$\langle\langle p_1{:}p_2,p_3\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}^{\leqslant k}\,\mathsf{m}_1\,]+\mathtt{P}[\,\mathtt{F}\,\mathsf{m}_{2,3}\,])$	4,4,8	3	158,609	1,282,435
$[e_{max},s_{max},k]$	6,4,6	3	944,727	9,071,885
	6,4,12	3	1,745,001	16,800,083
Power control	40,4,20	2	182,772	1,040,940
$\langle\langle p_1{:}p_2\rangle\rangle_{\max=?}(\mathtt{R}[\,\mathtt{F}\,e_1{=}0\,]+\mathtt{R}[\,\mathtt{C}^{\leqslant k}\,])$	80,4,20	2	917,988	5,153,700
$[e_{max},pow_{max},k]$	40,8,20	2	524,473	4,179,617
	80,8,20	2	3,806,240	28,950,948

Table 6: Model statistics for CSGs built verifying mixed nonzero-sum properties.

Case study & property [parameters]	Param. values	Inner Formula		Outer Formula	
		Val. iters.	Verif. time (s)	Val. iters.	Verif. time (s)
Robot coordination	4	12; 12	0.1	5	0.1
$\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ \phi\,]$	8	22; 22	5.0	10	0.8
$\phi{=}\langle\langle rbt_2\rangle\rangle\mathtt{R}_{\geqslant 10}[\,\mathtt{F}\,\mathsf{g}_2\,]$ $[l]$	12	31; 30	40.5	10	2.6
	16	40; 40	187.7	11	6.6
	24	58; 58	1,235	12	22.6
Robot coordination	4	8	0.1	24	0.1
$\langle\langle rbt_1, rbt_2\rangle\rangle\mathtt{P}_{\min=?}[\,\mathtt{F}\ \phi\,]$	8	17	20.8	18	1.4
$\phi{=}\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\min\leqslant 5}(\mathtt{R}[\,\mathtt{F}\,\mathsf{g}_1\,]+\mathtt{R}[\,\mathtt{F}\,\mathsf{g}_2\,])$ $[l]$	12	32	527.3	28	8.4
	16	39	4,664	37	24.9
Aloha	1,8	23	0.6	24	0.3
$\langle\langle usr_1, usr_2, usr_3\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ \phi\,]$	2,8	23	1.4	23	0.8
$\phi{=}\langle\langle usr_1{:}usr_2, usr_3\rangle\rangle_{\min\geqslant 2}(\mathtt{P}[\,\mathtt{F}\,\mathsf{s}_1\,]+\mathtt{P}[\,\mathtt{F}\,\mathsf{s}_{2,3}\,])$	3,8	22	7.9	23	2.8
$[b_{\max},D]$	4,8	22	39.8	23	9.3
	5,8	22	172.6	23	37.1

Table 7: Statistics for verification of nested properties for CSGs.

7.2 Case Studies

Next, we present more information about our case studies, to illustrate the applicability and benefits of our techniques. We use some of these examples to illustrate the benefits of concurrent stochastic games, in contrast to their turn-based counterpart; here, we build both TSG and CSG models for the case study and compare the results.

To study the benefits of nonzero-sum properties, we compare the results with corresponding zero-sum properties. For example, for a nonzero-sum formula of the form $\langle\langle C{:}C'\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\ \phi_1\,]+\mathtt{P}[\,\mathtt{F}\ \phi_2\,])$, we compute the value and an optimal strategy $\sigma_C^\star$ for coalition C of the formula $\langle\langle C\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ \phi_1\,]$, and then find the value of an optimal strategy for the coalition C' for $\mathtt{P}_{\min=?}[\,\mathtt{F}\ \phi_2\,]$ and $\mathtt{P}_{\max=?}[\,\mathtt{F}\ \phi_2\,]$ in the MDP induced by CSG when C follows $\sigma_C^\star$. The aim is to showcase the advantages of cooperation since, in many real-world applications, agents' goals are not strictly opposed and adopting a strategy that assumes antagonistic behaviour can have a negative impact from both individual and collective standpoints.

As will be seen, our results demonstrate that, by using nonzero-sum properties, at least one of the players gains and in almost all cases neither player loses (in the one case study where this is not the case, the gains far outweigh the losses). The individual SWNE/SCNE values for players need not be unique and, for all case

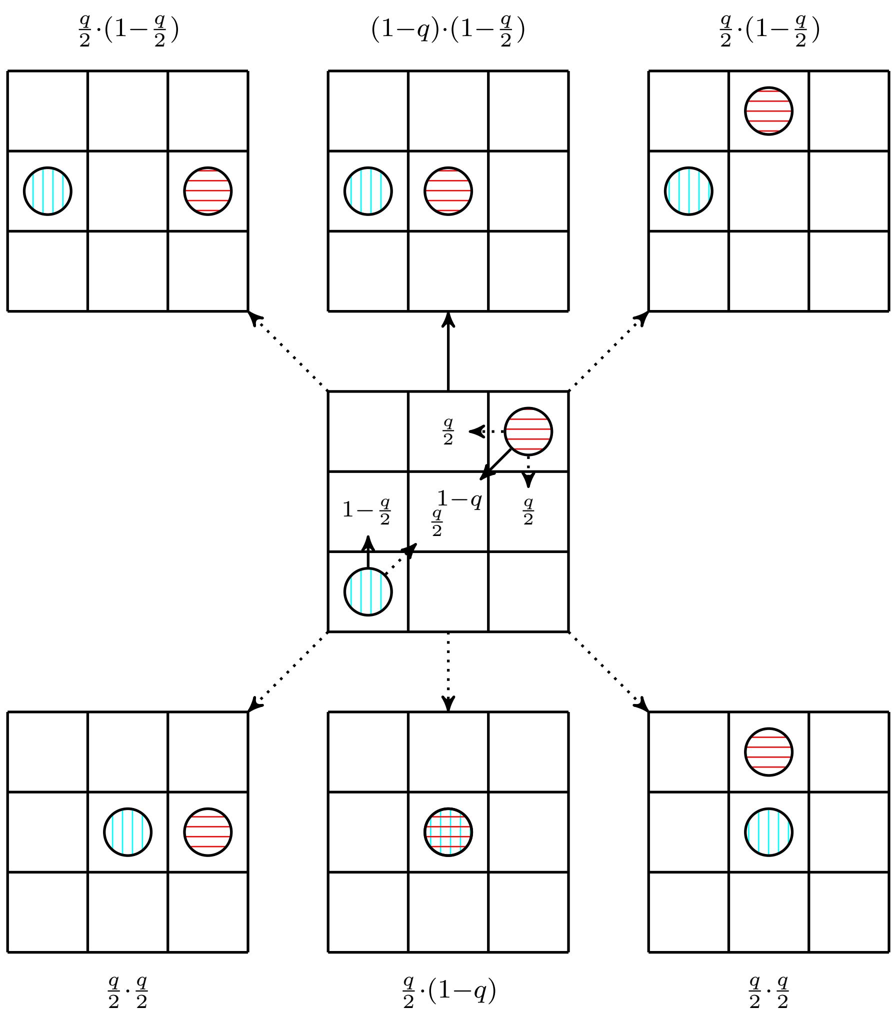

$\frac{q}{2}\cdot(1-\frac{q}{2})$
$(1-q)\cdot(1-\frac{q}{2})$
$\frac{q}{2}\cdot(1-\frac{q}{2})$
$\frac{q}{2}$
$1-\frac{q}{2}$
$1-q$
$\frac{q}{2}$
$\frac{q}{2}$
$\frac{q}{2}\cdot\frac{q}{2}$
$\frac{q}{2}\cdot(1-q)$
$\frac{q}{2}\cdot\frac{q}{2}$

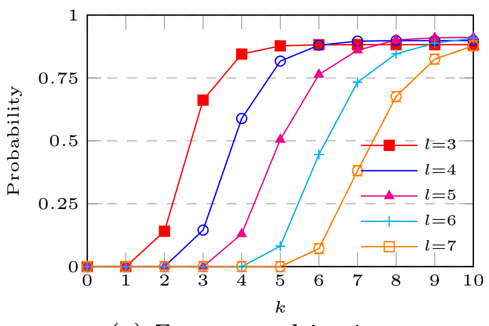
Fig. 3: Robot coordination on a 3×3 grid: probabilistic choices for one pair of action choices in the initial state. Solid lines indicate movement in the intended direction, dotted lines where there is deviation due to obstacles.

studies (except Aloha and medium access in which the players are not symmetric), the values can be swapped to give alternative SWNE/SCNE values.

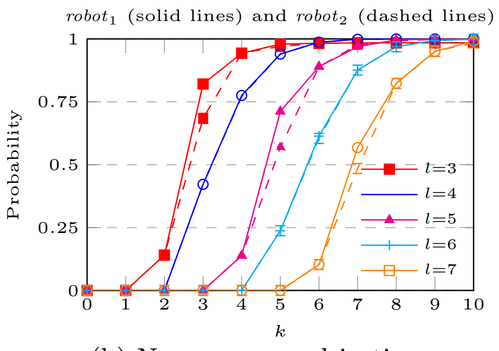
Finally, we note that, for infinite-horizon nonzero-sum properties, we compute the value of ε for the synthesised ε-NE and find that $\varepsilon = 0$ in all cases.

Robot Coordination. Our first case study concerns a scenario in which two robots move concurrently over a grid of size $l{\times}l$, briefly discussed in Example 5. The robots start in diagonally opposite corners and try to reach the corner from which the other starts. A robot can move either diagonally, horizontally or vertically towards its goal. Obstacles which hinder the robots as they move from location to location are modelled stochastically according to a parameter q (which we set to 0.25): when a robot moves, there is a probability that it instead moves in an adjacent direction, e.g., if it tries to move north west, then with probability $q/2$ it will instead move north and with the same probability west.

We can model this scenario as a two-player CSG, where the players correspond to the robots (rbt_1 and rbt_2), the states of the game represent their positions on the grid. In states where a robot has not reached its goal, it can choose between actions that move either diagonally, horizontally or vertically towards its goal (under the



(a) Zero-sum objectives



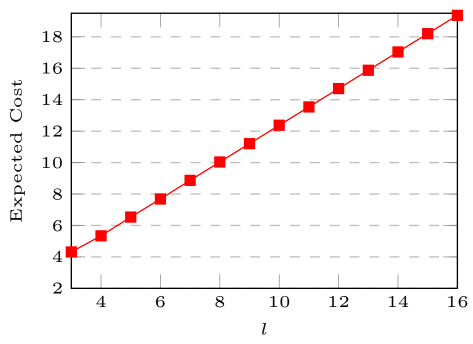
(b) Nonzero-sum objectives

Fig. 4: Robot coordination: probability of reaching the goal without crashing.

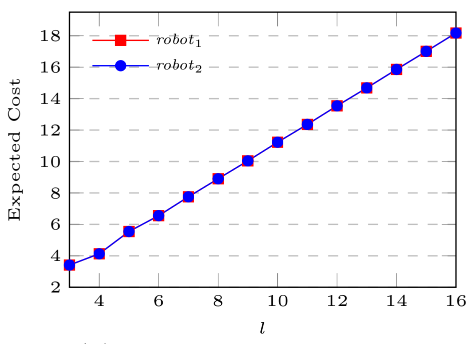
restriction that it remains in the grid after this move). For $i \in \{1,2\}$, we let goal_i be the atomic proposition labelling those states of the game in which rbt_i has reached its goal and crash the atomic proposition labelling the states in which the robots have crashed, i.e., are in the same grid location. In Figure 3, we present the states that can be reached from the initial state of the game when $l = 3$, when the robot in the south west corner tries to move north and the robot in the north east corner tries to move south west. As can be seen there are six different outcomes and the probability of the robots crashing is $\frac{q}{2}\cdot(1-q)$.

We first investigate the probability of the robots eventually reaching their goals without crashing for different size grids. In the zero-sum case, we find the values for the formula $\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\neg\mathsf{crash}\ \mathtt{U}\ \mathsf{goal}_1\,]$ converge to 1 as l increases; for example, the values for this formula in the initial states of game when $l = 5$, 10 and 20 are approximately 0.9116, 0.9392 and 0.9581, respectively. On the other hand, in the nonzero-sum case, considering SWNE values for the formula $\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max=?}(\mathtt{P}[\,\neg\mathsf{crash}\ \mathtt{U}\ \mathsf{goal}_1\,]{+}\mathtt{P}[\,\neg\mathsf{crash}\ \mathtt{U}\ \mathsf{goal}_2\,])$ and $l \geqslant 4$, we find that each robot can reach its goal with probability 1 (since time is not an issue, they can collaborate to avoid crashing).

We next consider the probability of the robots reaching their targets without crashing within a bounded number of steps. Figure 4 presents both the value for the (zero-sum) formula $\langle\langle rbt_1\rangle\rangle\mathtt{P}_{\max=?}[\,\neg\mathsf{crash}\ \mathtt{U}^{\leqslant k}\ \mathsf{goal}_1\,]$ and SWNE values for the formula $\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\max\geqslant 2}(\mathtt{P}[\,\neg\mathsf{crash}\ \mathtt{U}{\leqslant}\ k_1\ \mathsf{goal}_1\,]{+}\mathtt{P}[\,\neg\mathsf{crash}\ \mathtt{U}{\leqslant}\ k_2\ \mathsf{goal}_2\,])$, for a range of step bounds and grid sizes. When there is only one route to each goal within the bound (along the diagonal), i.e., when $k_1 = k_2 = l{-}1$, in the SWNE profile both robots take this route. In odd grids, there is a high chance of crashing, but also a chance one will deviate and the other reaches its goal. Initially, as the bound k increases, for odd grids the SWNE values for the robots are not equal (see Figure 4 right). Here, both robots following the diagonal does not yield a NE profile. First, the chance of crashing is high, and therefore the probability of the robots satisfying their objectives is low. Therefore it is advantageous for a robot to switch to a longer route as this will increase the probability of satisfying its objective, even taking into account that there is a greater chance it will run out of steps and changing its route will increase the probability of the other robot satisfying its objective by a greater amount (as the other robot will still be following the diagonal). Dually, both robots taking a longer route is not an NE profile, since if one robot switches to the diagonal route, then the probability of satisfying its objective will increase. It follows that, in a SWNE profile, one robot has to follow



(a) Zero-sum objectives



(b) Nonzero-sum objectives

Fig. 5: Robot coordination: expected steps to reach the goal.

the diagonal and the other take a longer route. As expected, if we compare the results, we see that the robots can improve their chances of reaching their goals by collaborating.

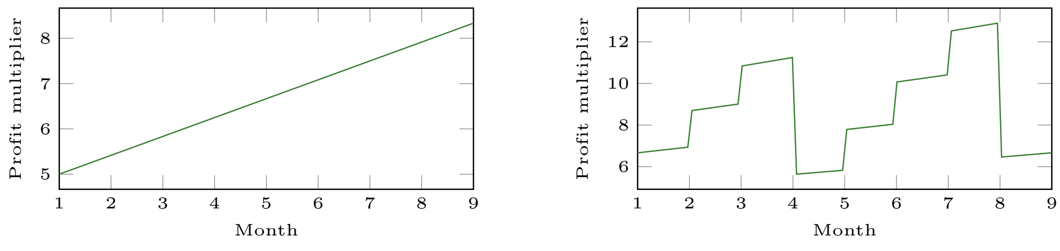
The next properties we consider concern the minimum expected number of steps for the robots to reach their goal. In Figure 5 we have plotted the values corresponding to the formula $\langle\langle rbt_2\rangle\rangle \mathtt{R}^{r_{steps}}_{\min=?}[\,\mathtt{F}\ \mathsf{goal}_2\,]$ and SCNE values for the individual players for $\langle\langle rbt_1{:}rbt_2\rangle\rangle_{\min=?}(\mathtt{R}^{r_{steps}}[\,\mathtt{F}\ \mathsf{goal}_1\,]{+}\mathtt{R}^{r_{steps}}[\,\mathtt{F}\ \mathsf{goal}_2\,])$ as the grid size l varies. The results again demonstrate that the players can gain by collaborating.

Futures market investors. This case study is a model of a futures market investor [56], which represents the interactions between investors and a stock market. For the TSG model of [56], in successive months, a single investor chooses whether to invest, next the market decides whether to bar the investor, with the restriction that the investor cannot be barred two months in a row or in the first month, and then the values of shares and a cap on values are updated probabilistically.

We have built and analysed several CSGs variants of the model, analysing optimal strategies for investors under adversarial conditions. First, we made a single investor and market take their decisions concurrently, and verified that this yielded no additional gain for the investor (see [81]). This is because the market and investor have the same information, and so the market knows when it is optimal for the investor to invest without needing to see its decision. We next modelled two competing investors who simultaneously decide whether to invest (and, as above, the market simultaneously decides which investors to bar). If the two investors cash in their shares in the same month, then their profits are reduced. We also consider several distinct profit models: 'normal market', 'later cash-ins', 'later cash-ins with fluctuation' and 'early cash-ins'. The first is from [56] and the remaining reward models either postponing cashing in shares or the early cashing in of shares. Figure 6 presents the 'later cash-ins' and 'later cash-ins with fluctuation' profit multipliers; see [81] for further details.

The CSG has 3 players: one for each investor and one representing the market who decides on the barring of investors. We study both the maximum profit of one investor and the maximum combined profit of both investors. For comparison, we also build a TSG model in which the investors first take turns to decide whether to invest (the ordering decided by the market) and then the market decides on whether to bar any of the investors.

Figure 7 shows the maximum expected value over a fixed number of months under the 'normal market' for both the profit of first investor and the combined



(a) Later cash-ins. (b) Later cash-ins with fluctuations.

Fig. 6: Futures market: payoff profiles.

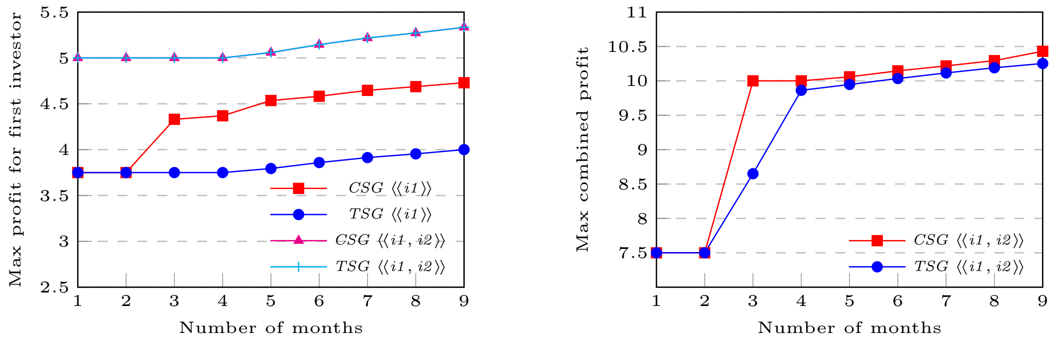


Fig. 7: Futures market investors: normal market.

profit of the two investors. For the former, we show results for the formulae $\langle\langle i_1\rangle\rangle \mathtt{R}^{profit_1}_{\max=?}[\,\mathtt{F}\ \mathsf{cashed\_in}_1\,]$, corresponding to the first investor acting alone, and $\langle\langle i_1, i_2\rangle\rangle \mathtt{R}^{profit_{1,2}}_{\max=?}[\,\mathtt{F}\ \mathsf{cashed\_in}_{1,2}\,]$ when in a coalition with the second investor. We plot the corresponding results from the TSG model for comparison. Figure 8 shows the maximum expected combined profit for the two 'later cash-ins' profiles. The variations in the combined profits of the investors for 'later cash-ins with fluctuations' are caused by the rise and fall in the profit multiplier under this profile, as shown in Figure 6.

When investors cooperate to maximise the profit of the first, results for the CSG and TSG models coincide. This follows from the discussion above since all the second investor can do is to make sure it does not invest at the same time as the first. For the remaining cases and given sufficient months, there is always a strategy in the concurrent setting that outperforms all turn-based strategies. The increase in profit for a single investor in the CSG model is due to the fact that, as the investors decisions are concurrent, the second cannot ensure it invests at the same time as the first, and hence decreases the profit of the first. In the case of combined profit, the difference arises because, although the market knows when it is optimal for one investor to invest, in the CSG model the market does not know which one will, and therefore may choose the wrong investor to bar.

We performed strategy synthesis to study the optimal actions of investors. By way of example, consider $\langle\langle i_1\rangle\rangle \mathtt{R}^{profit_1}_{\max=?}[\,\mathtt{F}\ \mathsf{cashed\_in}_1\,]$ over three months and for a normal market (see Figure 7 left). The optimal TSG strategy for the first investor is to invest in the first month (which the market cannot bar) ensuring an expected profit of 3.75. The optimal (randomised) CSG strategy is to invest:

- in the first month with probability $\sim$0.4949;
- in the second month with probability 1, if the second investor has cashed in;

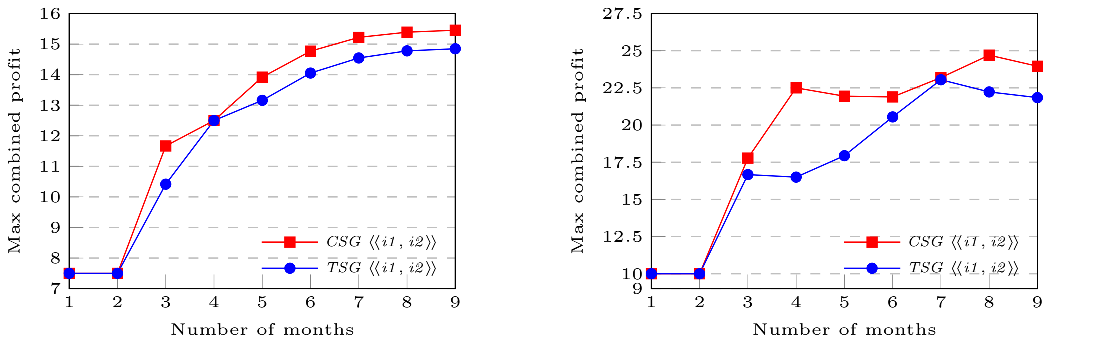


Fig. 8: Futures market: later cash-ins without (left) and with (right) fluctuations.

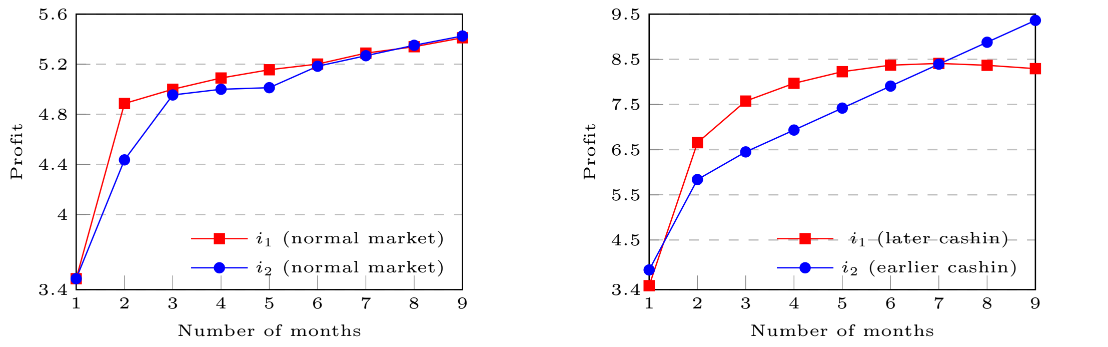


Fig. 9: Futures market: normal profiles (left) and mixed profiles (right) (p_{bar}=0.1).

– in the second month with probability ∼0.9649, if the second investor did not cash in at the end of the first month and the shares went up;
– in the second month with probability ∼0.9540, if the second investor did not cash in at the end of the first month and the shares went down;
– in the third month with probability 1 (this is the last month to invest).

Following this strategy, the first investor ensures an expected profit of ∼4.33.

We now make the market probabilistic, where, in any month when it did not bar the investor in the previous month (including the first), the probability that the market bars an individual investor equals p_{bar}. We consider nonzero-sum properties of the form $\langle\langle i_1{:}i_2\rangle\rangle_{\max=?}(\mathtt{R}^{profit_1}[\,\mathtt{F}\ \mathsf{cashed\_in}_1\,]+\mathtt{R}^{profit_2}[\,\mathtt{F}\ \mathsf{cashed\_in}_2\,])$, in which each investor tries to maximise their individual profit, for different reward structures. In Figures 9 and 10 we have plotted the results for the investors where the profit models of the investors follow a normal profile and where the profit models of the investors differ ('later cash-ins' for the first investor and 'early cash-ins' for second), when p_{bar} equals 0.1 and 0.5 respectively. The results demonstrate that, given more time and a more predictable market, i.e., when p_{bar} is lower, the players can collaborate to increase their profits.

Performing strategy synthesis, we find that the strategies in the mixed profiles model are for the investor with an 'early cash-ins' profit model to invest as soon as possible, i.e., it tries to invest in the first month and if this fails because it is barred, it will be able to invest in the second. On the other hand, for the investor with the 'later cash-ins' profile, the investor will delay investing until the chances of the shares failing start to increase or they reach the month before last and then invest (if the investor is barred in this month, they will be able to invest in the final month).

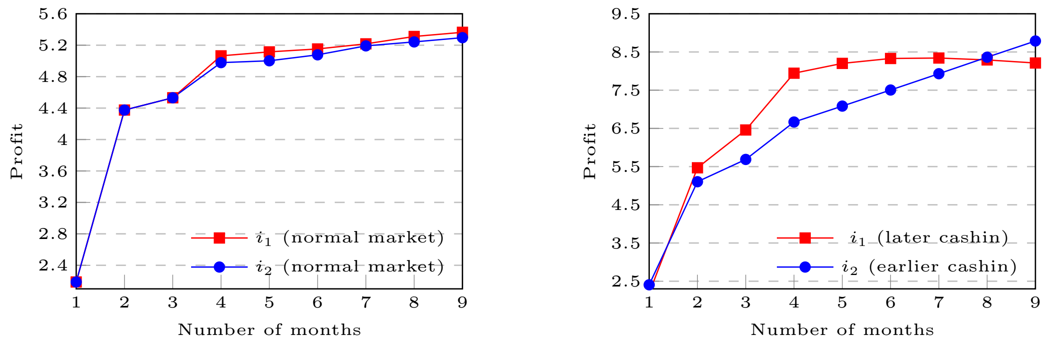


Fig. 10: Futures market: normal profiles (left) and mixed profiles (right) (p_{bar}=0.5).

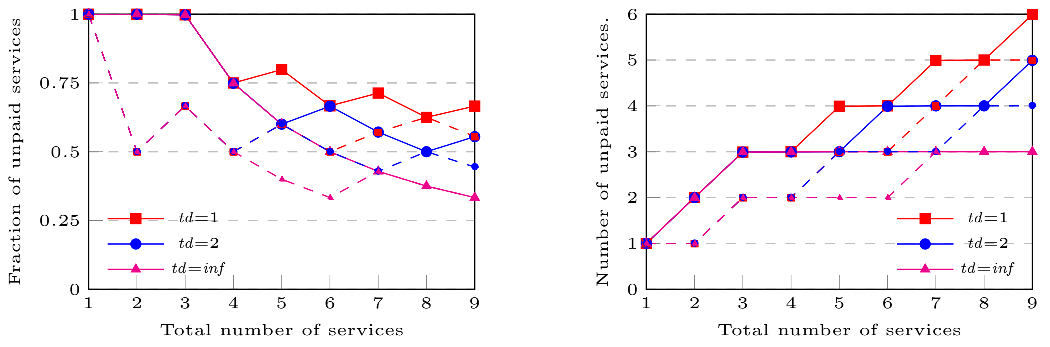


Fig. 11: User-centric network results (CSG/TSG values as solid/dashed lines).

Trust models for user-centric networks. Trust models for user-centric networks were analysed previously using TSGs in [50]. The analysis considered the impact of different parameters on the effectiveness of cooperation mechanisms between service providers. The providers share information on the measure of *trust* for users in a *reputation*-based setting. Each measure of trust is based on the service's previous interactions with the user (which previous services they paid for), and providers use this measure to block or allow the user to obtain services.

In the original TSG model, a single user can either make a request to one of three service providers or buy the service directly by paying maximum price. If the user makes a request to a service provider, then the provider decides to accept or deny the request based on the user's trust measure. If the request was accepted, the provider would next decide on the price again based on the trust measure, and the user would then decide whether to pay for the service and finally the provider would update its trust measure based on whether there was a payment. This sequence of steps would have to take place before any other interactions occurred between the user and other providers. Here we consider CSG models allowing the user to make requests and pay different service providers simultaneously and for the different providers to execute requests concurrently. There are 7 players: one for the user's interaction with each service provider, one for the user buying services directly and one for each of the 3 service providers. Three trust models were considered. In the first, the trust level was decremented by 1 ($td = 1$) when the user does not pay, decremented by 2 in the second ($td = 2$) and reset to 0 in the third ($td = inf$).

Figure 11 presents results for the maximum fraction and number of unpaid services the user can ensure for each trust model, corresponding to the formulae $\langle\langle usr \rangle\rangle \mathtt{R}^{ratio^-}_{\min=?}[\,\mathtt{F}\ \mathsf{finished}\,]$ and $\langle\langle usr \rangle\rangle \mathtt{R}^{unpaid^-}_{\min=?}[\,\mathtt{F}\ \mathsf{finished}\,]$ (to prevent not requesting any services and obtaining an infinite reward being the optimal choice of the user,

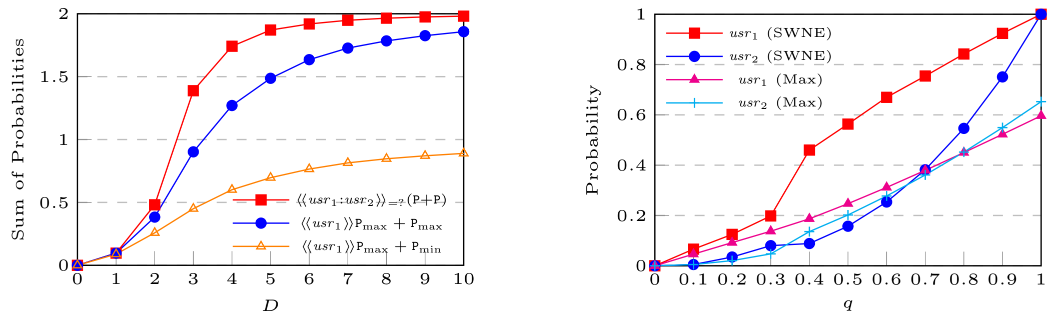


Fig. 12: Aloha: $\langle\langle usr_1{:}\{usr_2, usr_3\}\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\ (\mathsf{s}_1{\wedge}t{\leqslant}D)\,]{+}\mathtt{P}[\,\mathtt{F}\ (\mathsf{s}_2{\wedge}\mathsf{s}_3{\wedge}t{\leqslant}D)\,])$

we negate all rewards and find the minimum expected reward the user can ensure). The results for the original TSG model are included as dashed lines. The results demonstrate that the user can take advantage of the fact that in the CSG model it can request multiple services at the same time, and obtain more services without paying before the different providers get a chance to inform each other about non-payment. In addition, the results show that imposing a more severe penalty on the trust measure for non-payment reduces the number of services the user can obtain without paying.

Aloha. This case study concerns three users trying to send packets using the slotted ALOHA protocol. In a time slot, if a single user tries to send a packet, there is a probability (q) that the packet is sent; as more users try and send, then the probability of success decreases. If sending a packet fails, the number of slots a user waits before resending is set according to an exponential backoff scheme. More precisely, each user maintains a backoff counter which it increases each time there is a failure (up to $b_{\max}$) and, if the counter equals k, randomly chooses the slots to wait from $\{0, 1, \ldots, 2^k{-}1\}$.

We suppose that the three users are each trying to maximise the probability of sending their packet before a deadline D, with users 2 and 3 forming a coalition, which corresponds to the formula $\langle\langle usr_1{:}usr_2{,}usr_3\rangle\rangle_{\max=?}\mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_1 \wedge t{\leqslant}D)\,]+\mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_2 \wedge \mathsf{sent}_3 \wedge t{\leqslant}D)\,]$. Figure 12 presents total values as D varies (left) and individual values as q varies (right). Through synthesis, we find the collaboration is dependent on D and q. Given more time there is a greater chance for the users to collaborate by sending in different slots, while if q is large it is unlikely users need to repeatedly send, so again can send in different slots. As the coalition has more messages to send, their probabilities are lower. However, for the scenario with two users, the probabilities of the two users would still be different. In this case, although it is advantageous to initially collaborate and allow one user to try and send its first message, if the sending fails, given there is a bound on the time for the users to send, both users will try to send at this point as this is the best option for their individual goals.

We have also considered when the users try to minimise the expected time before their packets are sent, where users 2 and 3 form a coalition, represented by the formula $\langle\langle usr_1{:}usr_2{,}usr_3\rangle\rangle_{\min=?}(\mathtt{R}^{time}[\,\mathtt{F}\ \mathsf{sent}_1\,]{+}\mathtt{R}^{time}[\,\mathtt{F}\ (\mathsf{sent}_2 \wedge \mathsf{sent}_3)\,])$. When synthesising the strategies we see that the players collaborate with the coalition of users 2 and 3, letting user 1 to try and send before sending their messages. However, if user 1 fails to send, then the coalition either lets user 1 try again in case the user can do so immediately, and otherwise the coalition attempts to send their messages.

Finally, we have analysed when the players collaborate to maximise the probability of reaching a state where they can then send their messages with probability 1 within D time units (with users 2 and 3 in coalition), which is represented by the formula $\langle\langle usr_1{,}usr_2{,}usr_3\rangle\rangle\mathtt{P}_{\max=?}[\mathtt{F}\ \langle\langle usr_1{:}usr_2{,}usr_3\rangle\rangle_{\min\geqslant 2}\mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_1 \wedge t{\leqslant}D)\,] + \mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_2 \wedge \mathsf{sent}_3 \wedge t{\leqslant}D)\,]]$.

Intrusion detection policies. In [78], CSGs are used to model the interaction between an intrusion detection policy and attacker. The policy has a number of libraries it can use to detect attacks and the attacker has a number of different attacks which can incur different levels of damage if not detected. Furthermore, each library can only detect certain attacks. In the model, in each round the policy chooses a library to deploy and the attacker chooses an attack. A reward structure is specified representing the level of damage when an attack is not detected. The goal is to find optimal intrusion detection policies which correspond to finding a strategy for the policy that minimises damage, represented by synthesising a strategy for the formula $\langle\langle policy\rangle\rangle\mathtt{R}^{damage}_{\min=?}[\,\mathtt{C}^{\leqslant rounds}\,]$. We have constructed CSG models with two players (representing the policy and the attacker) for the two scenarios outlined in [78].

Jamming multi-channel radio systems. A CSG model for jamming multi-channel cognitive radio systems is presented in [79]. The system consists of a number of channels ($chans$), which can be in an occupied or idle state. The state of each channel remains fixed within a time slot and between slots is Markovian (i.e. the state changes randomly based only on the state of the channel in the previous slot). A secondary user has a subset of available channels and at each time-slot must decide which to use. There is a single attacker which again has a subset of available channels and at each time slot decides to send a jamming signal over one of them. The CSG has two players: one representing the secondary user and the other representing the attacker. Through the zero-sum property $\langle\langle user\rangle\rangle\mathtt{P}_{\max=?}[\,\mathtt{F}\ (sent \geqslant slots/2)\,]$ we find the optimal strategy for the secondary user to maximize the probability that at least half their messages are sent against any possible attack. We have also considered the expected number of messages sent by the kth time-slot: $\langle\langle user\rangle\rangle\mathtt{R}^{sent}_{\max=?}[\,\mathtt{I}^{=k}\,]$.

Medium Access Control. This case study extends the CSG model from Example 4 to three users and assumes that the probability of a successful transmission is dependent on the number of users that try and send ($q_1 = 0.95$, $q_2 = 0.75$ and $q_3 = 0.5$). The energy of each user is bounded by $e_{\max}$. We suppose the first user acts in isolation and the remaining users form a coalition. The first nonzero-sum property we consider is $\langle\langle p_1{:}p_2{,}p_3\rangle\rangle_{\max=?}(\mathtt{R}^{sent_1}[\,\mathtt{C}^{\leqslant k_1}\,]{+}\mathtt{R}^{sent_{2,3}}[\,\mathtt{C}^{\leqslant k_2}\,])$, which corresponds to each coalition trying to maximise the expected number of messages they send over a bounded number of steps. On the other hand, the second property is $\langle\langle p_1{:}p_2{,}p_3\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}^{\leqslant k}\ (mess_1 = s_{\max})\,]{+}\mathtt{P}[\,\mathtt{F}\ (mess_2{+}mess_3 = 2{\cdot}s_{\max})\,])$ and here the coalitions try to maximise the probability of successfully transmitting a certain number of messages ($s_{\max}$ for the first user and $2{\cdot}s_{\max}$ for the coalition of the second and third users), where in addition the first user has to do this in a bounded number of steps (k).

Power Control. Our final case study is based on a model of power control in cellular networks from [14]. In the model, phones emit signals over a cellular network and the signals can be strengthened by increasing the power level up to a bound

($pow_{\max}$). A stronger signal can improve transmission quality, but uses more energy and lowers the quality of other transmissions due to interference. We extend this model by adding a failure probability (q_{fail}) when a power level is increased and assume each phone has a limited battery capacity ($e_{\max}$). Based on [14], we associate a reward structure with each phone representing transmission quality dependent both on its power level and that of other phones due to interference. We consider the nonzero-sum property $\langle\langle p_1{:}p_2 \rangle\rangle_{\max=?}(\mathtt{R}^{r_1}[\,\mathtt{F}\ (e_1=0)\,]{+}\mathtt{R}^{r_2}[\,\mathtt{F}\ (e_2=0)\,])$, where each user tries to maximise their expected reward before their phone's battery is empty. We have also analysed the properties: $\langle\langle p_1{:}p_2 \rangle\rangle_{\max=?}(\mathtt{R}^{r_1}[\,\mathtt{F}\ (e_1=0)\,] + \mathtt{R}^{r_2}[\,\mathtt{C}^{\leqslant k}\,])$, where the objective of the second user is to instead maximise their expected reward over a bounded number of steps (k), and $\langle\langle p_1{:}p_2 \rangle\rangle_{\max=?}(\mathtt{R}^{r_1}[\,\mathtt{I}^{=k_1}\,] + \mathtt{R}[\,\mathtt{I}^{=k_2}\,])$, where the objective of user i is to maximise their reward at the k_ith step.

8 Conclusions

In this paper, we have designed and implemented an approach for the automatic verification of a large subclass of CSGs. We have extended the temporal logic rPATL to allow for the specification of equilibria-based (nonzero-sum) properties, where two players or coalitions with distinct goals can collaborate. We have then proposed and implemented algorithms for verification and strategy synthesis using this extended logic, including both zero-sum and nonzero-sum properties, in the PRISM-games model checker. In the case of finite-horizon properties the algorithms are exact, while for infinite-horizon they are approximate using value iteration. We have also extended the PRISM-games modelling language, adding new features tailored to CSGs. Finally, we have evaluated the approach on a range of case studies that have demonstrated the benefits of CSG models compared to TSGs and of nonzero-sum properties as a means to synthesise strategies that are collectively more beneficial for all players in a game.

The main challenge in implementing the model checking algorithms is efficiently solving matrix and bimatrix games at each state in each step of value iteration for zero-sum and nonzero-sum properties, respectively, which are non-trivial optimisation problems. For bimatrix games, this furthermore requires finding an optimal equilibrium, which currently relies on iteratively restricting the solution search space. Solution methods can be sensitive to floating-point arithmetic issues, particularly for bimatrix games; arbitrary precision representations may help here to alleviate these problems.

There are a number of directions for future work. First, we plan to consider additional properties such as multi-objective queries. We are also working on extending the implementation to consider alternative solution methods (e.g., policy iteration and using CPLEX [40] to solve matrix games) and a symbolic (binary decision diagram based) implementation and other techniques for Nash equilibria synthesis such as an MILP-based solution using regret minimisation. Lastly, we are considering extending the approach to partially observable strategies, multi-coalitional games, building on [47], and mechanism design.

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

References

1. de Alfaro, L.: Computing minimum and maximum reachability times in probabilistic systems. In: J. Baeten, S. Mauw (eds.) Proc. CONCUR'99, *LNCS*, vol. 1664, pp. 66–81. Springer (1999)
2. de Alfaro, L., Henzinger, T.: Concurrent omega-regular games. In: LICS'00, pp. 141–154. ACM (2000)
3. de Alfaro, L., Henzinger, T., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science **386**(3), 188–217 (2007)
4. de Alfaro, L., Majumdar, R.: Quantitative solution of omega-regular games. Journal of Computer and System Sciences **68**(2), 374–397 (2004)
5. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49**(5), 672–713 (2002)
6. Arslan, G., Yüksel, S.: Distributionally consistent price taking equilibria in stochastic dynamic games. In: Proc. CDC'17, pp. 4594–4599. IEEE (2017)
7. Ashok, P., Chatterjee, K., Kretínský, J., Weininger, M., Winkler, T.: Approximating values of generalized-reachability stochastic games. In: Proc. LICS'20, pp. 102–115. ACM (2020)
8. Avis, D., Rosenberg, G., Savani, R., von Stengel, B.: Enumeration of Nash equilibria for two-player games. Economic Theory **42**(1), 9–37 (2010)
9. Basset, N., Kwiatkowska, M., Wiltsche, C.: Compositional strategy synthesis for stochastic games with multiple objectives. Information and Computation **261**(3), 536–587 (2018)
10. Bianco, A., de Alfaro, L.: Model checking of probabilistic and nondeterministic systems. In: P. Thiagarajan (ed.) Proc. FSTTCS'95, *LNCS*, vol. 1026, pp. 499–513. Springer (1995)
11. Bouyer, P., Markey, N., Stan, D.: Mixed Nash equilibria in concurrent games. In: V. Raman, S. Suresh (eds.) Proc. FSTTCS'14, *LIPICS*, vol. 29, pp. 351–363. Leibniz-Zentrum für Informatik (2014)
12. Bouyer, P., Markey, N., Stan, D.: Stochastic equilibria under imprecise deviations in terminal-reward concurrent games. In: D. Cantone, G. Delzanno (eds.) Proc. GandALF'16, *EPTCS*, vol. 226, pp. 61–75. Open Publishing Association (2016)
13. Brázdil, T., Chatterjee, K., Chmelík, M., Forejt, V., Křetínský, J., Kwiatkowska, M., Parker, D., Ujma, M.: Verification of Markov decision processes using learning algorithms. In: F. Cassez, J.F. Raskin (eds.) Proc. ATVA'14, *LNCS*, vol. 8837, pp. 98–114. Springer (2014)
14. Brenguier, R.: PRALINE: A tool for computing Nash equilibria in concurrent games. In: N. Sharygina, H. Veith (eds.) Proc. CAV'13, *LNCS*, vol. 8044, pp. 890–895. Springer (2013). lsv.fr/Software/praline/
15. Brihaye, T., Bruyère, V., Goeminne, A., Raskin, J.F., van den Bogaard, M.: The complexity of subgame perfect equilibria in quantitative reachability games. In: W. Fokkink, R. van Glabbeek (eds.) Proc. CONCUR'19, *LIPIcs*, vol. 140, pp. 13:1–13:16. Leibniz-Zentrum für Informatik (2019)
16. Čermák, P., Lomuscio, A., Mogavero, F., Murano, A.: MCMAS-SLK: A model checker for the verification of strategy logic specifications. In: A. Biere, R. Bloem (eds.) Proc. CAV'14, *LNCS*, vol. 8559, pp. 525–532. Springer (2014)
17. Chatterjee, K.: Stochastic ω-regular games. Ph.D. thesis, University of California at Berkeley (2007)
18. Chatterjee, K., de Alfaro, L., Henzinger, T.: Strategy improvement for concurrent reachability and turn-based stochastic safety games. Journal of Computer and System Sciences **79**(5), 640–657 (2013)
19. Chatterjee, K., Henzinger, T.: Value iteration. In: O. Grumberg, H. Veith (eds.) 25 Years of Model Checking, *LNCS*, vol. 5000, pp. 107–138. Springer (2008)
20. Chatterjee, K., Henzinger, T.: A survey of stochastic ω-regular games. Journal of Computer and System Sciences **78**(2), 394–413 (2012)
21. Chatterjee, K., Henzinger, T., Jobstmann, B., Radhakrishna, A.: GIST: A solver for probabilistic games. In: T. Touili, B. Cook, P. Jackson (eds.) Proc. CAV'10, *LNCS*, vol. 6174, pp. 665–669. Springer (2010). pub.ist.ac.at/gist/

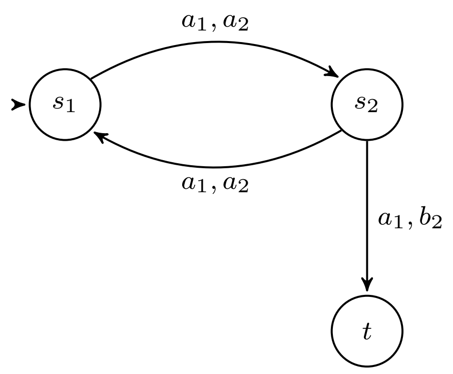
22. Chatterjee, K., Majumdar, R., Jurdziński, M.: On Nash equilibria in stochastic games. In: J. Marcinkowski, A. Tarlecki (eds.) Proc. CSL'04, *LNCS*, vol. 3210, pp. 26–40. Springer (2004)
23. Chen, T., Forejt, V., Kwiatkowska, M., Parker, D., Simaitis, A.: Automatic verification of competitive stochastic systems. Formal Methods in System Design **43**(1), 61–92 (2013)
24. Chen, T., Forejt, V., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: On stochastic games with multiple objectives. In: K. Chatterjee, J. Sgall (eds.) Proc. MFCS'13, *LNCS*, vol. 8087, pp. 266–277. Springer (2013)
25. Cheng, C., Knoll, A., Luttenberger, M., Buckl, C.: GAVS+: An open platform for the research of algorithmic game solving. In: P. Abdulla, K. Leino (eds.) Proc. TACAS'11, *LNCS*, vol. 6605, pp. 258–261. Springer (2011). sourceforge.net/projects/gavsplus/
26. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: C. Ramakrishnan, J. Rehof (eds.) Proc. TACAS'08, *LNCS*, vol. 4963, pp. 337–340. Springer (2008). github.com/Z3Prover/z3
27. Dileepa, F., Dong, N., Jegourel, C., Dong, J.: Verification of strong Nash-equilibrium for probabilistic BAR systems. In: J. Sun, M. Sun (eds.) Proc. ICFEM'18, *LNCS*, vol. 11232, pp. 106–123. Springer (2018)
28. Dutertre, B.: Yices 2.2. In: A. Biere, R. Bloem (eds.) Proc CAV'14, *LNCS*, vol. 8559, pp. 737–744. Springer (2014). yices.csl.sri.com
29. Fearnley, J., Savani, R.: The complexity of the simplex method. In: Proc. STOC'15, pp. 201–208. ACM (2015)
30. Fudenberg, D., Levine, D.: Subgame-perfect equilibria of finite- and infinite-horizon games. Journal of Economic Theory **31**(2), 251–268 (1983)
31. Gansner, E., Koutsofios, E., North, S.: Drawing graphs with Dot (2015). Dot User's Manual
32. Gilboa, I., Zemel, E.: Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior **1**(1), 80–93 (1989)
33. Gutierrez, J., Najib, M., Giuseppe, P., Wooldridge, M.: Equilibrium design for concurrent games. In: W. Fokkink, R. van Glabbeek (eds.) Proc. CONCUR'19, *LIPIcs*, vol. 140, pp. 22:1–22:16. Leibniz-Zentrum für Informatik (2019)
34. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.: EVE: A tool for temporal equilibrium analysis. In: S. Lahiri, C. Wang (eds.) Proc. ATVA'18, *LNCS*, vol. 11138, pp. 551–557. Springer (2018). github.com/eve-mas/eve-parity
35. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.: Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. Artificial Intelligence **287**(103353) (2020)
36. Haddad, S., Monmege, B.: Interval iteration algorithm for MDPs and IMDPs. Theoretical Computer Science **735**, 111–131 (2018)
37. Hansen, K., Ibsen-Jensen, R., Miltersen, P.: The complexity of solving reachability games using value and strategy iteration. Theory of Computing Systems **55**, 380–403 (2011)
38. Hansson, H., Jonsson, B.: A logic for reasoning about time and reliability. Formal Aspects of Computing **6**(5), 512–535 (1994)
39. Hilbe, C., Šimsa, Š., Chatterjee, K., Nowak, M.: Evolution of cooperation in stochastic games. Nature **559** (2018)
40. ILOG CPLEX. ibm.com/products/ilog-cplex-optimization-studio
41. Karmarkar, N.: A new polynomial-time algorithm for linear programming. Combinatorica **4**(4), 373–395 (1984)
42. Kelmendi, E., Krämer, J., Kretínský, J., Weininger, M.: Value iteration for simple stochastic games: Stopping criterion and learning algorithm. In: H. Chockler, G. Weissenbacher (eds.) Proc. CAV'18, *LNCS*, vol. 10981, pp. 623–642. Springer (2018)
43. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. Springer (1976)
44. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: G. Gopalakrishnan, S. Qadeer (eds.) Proc CAV'11, *LNCS*, vol. 6806, pp. 585–591. Springer (2011). prismmodelchecker.org
45. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Automated verification of concurrent stochastic games. In: A. Horvath, A. McIver (eds.) Proc. QEST'18, *LNCS*, vol. 11024, pp. 223–239. Springer (2018)
46. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Equilibria-based probabilistic model checking for concurrent stochastic games. In: M. ter Beek, A. McIver, J. Oliveira (eds.) Proc. FM'19, *LNCS*, vol. 11800, pp. 298–315. Springer (2019)

47. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Multi-player equilibria verification for concurrent stochastic games. In: M. Gribaudo, D. Jansen, A. Remke (eds.) Proc. QEST'20, LNCS. Springer (2020). To appear
48. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In: Proc. 32nd International Conference on Computer Aided Verification (CAV'20), *LNCS*, vol. 12225, pp. 475–487. Springer (2020)
49. Kwiatkowska, M., Parker, D.: Automated verification and strategy synthesis for probabilistic systems. In: D.V. Hung, M. Ogawa (eds.) Proc. ATVA'13, *LNCS*, vol. 8172, pp. 5–22. Springer (2013)
50. Kwiatkowska, M., Parker, D., Simaitis, A.: Strategic analysis of trust models for user-centric networks. In: F. Mogavero, A. Murano, M. Vardi (eds.) Proc. SR'13, *EPTCS*, vol. 112, pp. 53–60. Open Publishing Association (2013)
51. Kwiatkowska, M., Parker, D., Wiltsche, C.: PRISM-games: Verification and strategy synthesis for stochastic multi-player games with multiple objectives. Software Tools for Technology Transfer **20**(2), 195–210 (2018)
52. Lemke, C., J. Howson, J.: Equilibrium points of bimatrix games. Journal of the Society for Industrial and Applied Mathematics **12**(2), 413–423 (1964)
53. Lozovanu, D., Pickl, S.: Determining Nash equilibria for stochastic positional games with discounted payoffs. In: J. Rothe (ed.) Proc. ADT'17, *LNAI*, vol. 10576, pp. 339–343. Springer (2017)
54. LPSolve (version 5.5). lpsolve.sourceforge.net/5.5/
55. Martin, D.: The determinacy of Blackwell games. Journal of Symbolic Logic **63**(4), 1565–1581 (1998)
56. McIver, A., Morgan, C.: Results on the quantitative mu-calculus qMu. ACM Trans. Computational Logic **8**(1) (2007)
57. McKelvey, R., McLennan, A., Turocy, T.: Gambit: Software tools for game theory. gambit-project.org
58. Nash, J.: Equilibrium points in n-person games. Proc. Natl. Acad. Sci **36**, 48–49 (1950)
59. von Neumann, J.: Zur theorie der gesellschaftsspiele. Mathematische Annalen **100**, 295–320 (1928)
60. von Neumann, J., Morgenstern, O., Kuhn, H., Rubinstein, A.: Theory of Games and Economic Behavior. Princeton University Press (1944)
61. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Algorithmic Game Theory. Cambridge University Press (2007)
62. Nudelman, E., Wortman, J., Shoham, Y., Leyton-Brown, K.: Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In: Proc. AAMAS'04, pp. 880–887. IEEE (2004). gamut.stanford.edu
63. Osborne, M., Rubinstein, A.: An Introduction to Game Theory. Oxford University Press (2004)
64. Pacheco, J., Santos, F., Souza, M., Skyrms, B.: Evolutionary dynamics of collective action. In: The Mathematics of Darwin's Legacy, Mathematics and Biosciences in Interaction, pp. 119–138. Springer (2011)
65. Papadimitriou, C.: On the complexity of the parity argument and other inefficient proofs of existence. Journal of Computer and System Sciences **48**(3), 498–532 (1994)
66. Porter, R., Nudelman, E., Shoham, Y.: Simple search methods for finding a Nash equilibrium. In: Proc. AAAI'04, pp. 664–669. AAAI Press (2004)
67. Prasad, H., Prashanth, L., Bhatnagar, S.: Two-timescale algorithms for learning Nash equilibria in general-sum stochastic games. In: Proc. AAMAS'15, pp. 1371–1379. IFAAMAS (2015)
68. Puterman, M.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley and Sons (1994)
69. Qin, H., Tang, W., Tso, R.: Rational quantum secret sharing. Scientific Reports **8**(11115) (2018)
70. Raghavan, T., Filar, J.: Algorithms for stochastic games — a survey. Zeitschrift für Operations Research **35**(6), 437–472 (1991)
71. Sandholm, T., Gilpin, A., Conitzer, V.: Mixed-integer programming methods for finding Nash equilibria. In: Proc. AAAI'05, pp. 495–501. AAAI Press (2005)
72. Schwalbe, U., Walker, P.: Zermelo and the early history of game theory. Games and Economic Behavior **34**(1), 123–137 (2001)
73. Shapley, L.: Stochastic games. PNAS **39**, 1095–1100 (1953)

74. Shapley, L.: A note on the Lemke-Howson algorithm. In: M. Balinski (ed.) Pivoting and Extension: In honor of A.W. Tucker, *Mathematical Programming Studies*, vol. 1, pp. 175–189. Springer (1974)
75. Todd, M.: The many facets of linear programming. Mathematical Programming **91**(3), 417–436 (2002)
76. Toumi, A., Gutierrez, J., Wooldridge, M.: A tool for the automated verification of Nash equilibria in concurrent games. In: M. Leucker, C. Rueda, F. Valencia (eds.) Proc. ICTAC'15, *LNCS*, vol. 9399, pp. 583–594. Springer (2015)
77. Ummels, M.: Stochastic multiplayer games: Theory and algorithms. Ph.D. thesis, RWTH Aachen University (2010)
78. Zhu, Q., Başar, T.: Dynamic policy-based IDS configuration. In: CDC'09, pp. 8600–8605. IEEE (2009)
79. Zhu, Q., Li, H., Han, Z., Başar, T.: A stochastic game model for jamming in multi-channel cognitive radio systems. In: Proc. ICC'10, pp. 1–6. IEEE (2010)
80. PRISM-games web site. prismmodelchecker.org/games/
81. Supporting material. prismmodelchecker.org/subm/fmsd-csgs/

A Convergence of Zero-Sum Reachability Reward Formulae

In this appendix we give a witness to the failure of convergence for value iteration when verifying zero-sum formulae with an infinite horizon reward objective if Assumption 1 does not hold.



$$r_A(s,(c_1,c_2)) = \begin{cases} -1 & \text{if } s = s_1 \text{ and } (c_1,c_2) = (a_1,a_2) \\ 1 & \text{if } s = s_2 \text{ and } (c_1,c_2) = (a_1,a_2) \\ 0 & \text{otherwise} \end{cases}$$

$$r_S(s) = \ 0 \text{ for all } s$$

Fig. 13: Counterexample for zero-sum expected reachability reward properties.

Consider the CSG in Figure 13 with players p_1 and p_2 and the zero-sum state formula $\phi = \langle\langle p_1, p_2 \rangle\rangle \mathtt{R}^{r}_{\max=?}[\,\mathtt{F}\ \mathsf{a}\,]$, where a is the atomic proposition satisfied only by state t. Clearly, state s_1 does not reach either the target of the formula or an absorbing state with probability 1 under all strategy profiles, while the reward for the state-action pair $(s_1,(a_1,a_2))$ is negative. Applying the value iteration algorithm of Section 4, we see that the values for state s_1 oscillate between 0 and -1, while the values for state s_2 oscillate between 0 and 1.

B Convergence of Nonzero-Sum Probabilistic Reachability Properties

In this appendix we give a witness to the failure of convergence for value iteration when verifying nonzero-sum formulae with infinite horizon probabilistic objectives if Assumption 2 does not hold.
Consider the CSG in Figure 14 with players p_1 and p_2 (an adaptation of a TSG example from [12]) and the nonzero-sum state formula $\langle\langle p_1{:}p_2 \rangle\rangle_{\max=?}(\theta)$, where $\theta = \mathtt{P}[\,\mathtt{F}\ \mathsf{a}_1\,]{+}\mathtt{P}[\,\mathtt{F}\ \mathsf{a}_2\,]$ and a_i is the atomic proposition satisfied only by the state t_i. Clearly, this CSG has a non-terminal end component as one can remain in $\{s_1, s_2\}$ indefinitely or leave at any time.

Applying the value iteration algorithm of Section 4, we have:

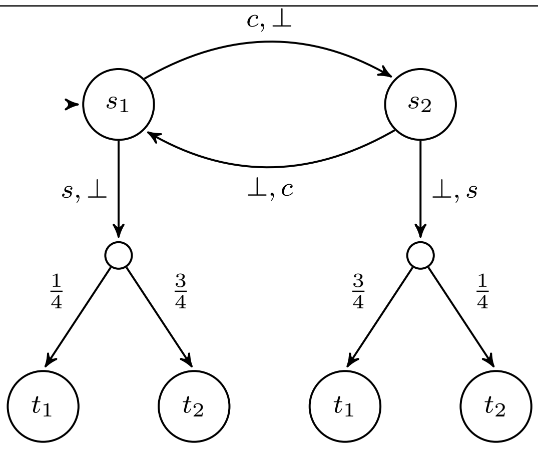


Fig. 14: Counterexample for nonzero-sum probabilistic reachability properties.

- In the first iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 1)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} 0 \\ \frac{1}{4} \end{pmatrix} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} 0 \\ \frac{3}{4} \end{pmatrix}$$

  i.e. the values $(\frac{1}{4}, \frac{3}{4})$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 1)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} 0 & \frac{3}{4} \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} 0 & \frac{1}{4} \end{pmatrix} \end{array}$$

  i.e. the values $(\frac{3}{4}, \frac{1}{4})$.
- In the second iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 2)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{3}{4} \\ \frac{1}{4} \end{pmatrix} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}$$

  i.e. the values $(\frac{3}{4}, \frac{1}{4})$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 2)$ are the SWNE values of the bimatrix games:

$$\mathsf{Z}_1 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{1}{4} & \frac{3}{4} \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{3}{4} & \frac{1}{4} \end{pmatrix} \end{array}$$

  i.e. the values $(\frac{1}{4}, \frac{3}{4})$.
- In the third iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 3)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{1}{4} \\ \frac{1}{4} \end{pmatrix} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{3}{4} \\ \frac{3}{4} \end{pmatrix}$$

  i.e. the values $(\frac{1}{4}, \frac{3}{4})$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 3)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{3}{4} & \frac{3}{4} \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{1}{4} & \frac{1}{4} \end{pmatrix} \end{array}$$

  i.e. the values $(\frac{3}{4}, \frac{1}{4})$.
- In the fourth iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 4)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{3}{4} \\ \frac{1}{4} \end{pmatrix} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{cc} & \perp \\ c & \\ s & \end{array}\!\!\begin{pmatrix} \frac{1}{4} \\ \frac{3}{4} \end{pmatrix}$$

i.e. the values $(\frac{3}{4}, \frac{1}{4})$, and $\mathtt{V}_{\mathsf{G}^C}(s_2, \theta, 4)$ are the SWNE values of the bimatrix game:

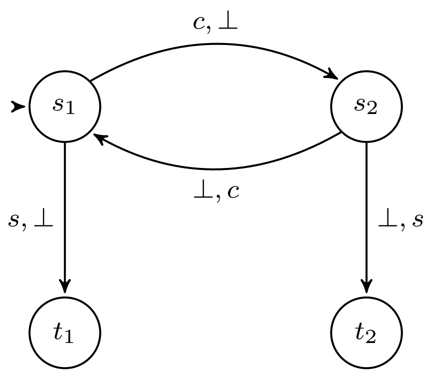
$$\mathsf{Z}_1 = \begin{array}{c|cc} & c & s \\ \perp & \frac{3}{4} & \frac{3}{4} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c|cc} & c & s \\ \perp & \frac{1}{4} & \frac{1}{4} \end{array}$$

i.e. the values $(\frac{3}{4}, \frac{1}{4})$.

As can be seen the values computed at each iteration for the states s_1 and s_2 will oscillate between $(\frac{1}{4}, \frac{3}{4})$ and $(\frac{3}{4}, \frac{1}{4})$.

C Convergence of Nonzero-Sum Expected Reachability Properties

In this appendix we give a witness to the failure of convergence for value iteration when verifying nonzero-sum formulae with infinite horizon reward objectives if Assumption 3 does not hold.



$$r^1_A(s,(a_1,a_2)) = \begin{cases} \frac{1}{3} & \text{if } s = s_1 \text{ and } (a_1,a_2) = (s,\perp) \\ 2 & \text{if } s = s_2 \text{ and } (a_1,a_2) = (\perp,s) \\ 0 & \text{otherwise} \end{cases}$$

$$r^1_S(s) = 0 \text{ for all } s$$

$$r^2_A(s,(a_1,a_2)) = \begin{cases} 1 & \text{if } s = s_1 \text{ and } (a_1,a_2) = (s,\perp) \\ \frac{1}{3} & \text{if } s = s_2 \text{ and } (a_1,a_2) = (\perp,s) \\ 0 & \text{otherwise} \end{cases}$$

$$r^2_S(s) = 0 \text{ for all } s$$

Fig. 15: Counterexample for nonzero-sum expected reachability properties.

Consider the CSG in Figure 15 with players p_1 and p_2 (which again is an adaptation of a TSG example from [12]) and the nonzero-sum state formula $\langle\langle p_1{:}p_2\rangle\rangle_{\max=?}(\theta)$, where $\theta = \mathtt{R}^{r_1}[\,\mathtt{F}\ \mathtt{a}\,]{+}\mathtt{R}^{r_2}[\,\mathtt{F}\ \mathtt{a}\,]$ and $\mathtt{a}$ is the atomic proposition satisfied only by the states t_1 and t_2. Clearly, there are strategy profiles for which the targets are not reached with probability 1.

Applying the value iteration algorithm of Section 4, we have:

– In the first iteration $\mathtt{V}_{\mathsf{G}^C}(s_1, \theta, 1)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c|c} & \perp \\ c & 0 \\ s & \frac{1}{3} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c|c} & \perp \\ c & 0 \\ s & 1 \end{array}$$

i.e. the values $(\frac{1}{3}, 1)$, and $\mathtt{V}_{\mathsf{G}^C}(s_2, \theta, 1)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c|cc} & c & s \\ \perp & 0 & 2 \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c|cc} & c & s \\ \perp & 0 & \frac{1}{3} \end{array}$$

i.e. the values $(2, \frac{1}{3})$.

– In the second iteration $\mathtt{V}_{\mathsf{G}^C}(s_1, \theta, 2)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c|c} & \perp \\ c & 2 \\ s & \frac{1}{3} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c|c} & \perp \\ c & \frac{1}{3} \\ s & 1 \end{array}$$

i.e. the values $(2, \frac{1}{3})$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 2)$ are the SWNE values of the bimatrix games:

$$\mathsf{Z}_1 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{1}{3} & 2 \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} 1 & \frac{1}{3} \end{pmatrix} \end{array}$$

i.e. the values $(\frac{1}{3}, 1)$.

– In the third iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 3)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c c} & \perp \\ \begin{array}{c} c \\ s \end{array} & \begin{pmatrix} \frac{1}{3} \\ \frac{1}{3} \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c c} & \perp \\ \begin{array}{c} c \\ s \end{array} & \begin{pmatrix} 1 \\ 1 \end{pmatrix} \end{array}$$

i.e. the values $(\frac{1}{3}, 1)$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 3)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} 2 & 2 \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{1}{3} & \frac{1}{3} \end{pmatrix} \end{array}$$

i.e. the values $(2, \frac{1}{3})$.

– In the fourth iteration $\mathsf{V}_{\mathsf{G}^C}(s_1, \theta, 4)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c c} & \perp \\ \begin{array}{c} c \\ s \end{array} & \begin{pmatrix} 2 \\ \frac{1}{3} \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c c} & \perp \\ \begin{array}{c} c \\ s \end{array} & \begin{pmatrix} \frac{1}{3} \\ 1 \end{pmatrix} \end{array}$$

i.e. the values $(2, \frac{1}{3})$, and $\mathsf{V}_{\mathsf{G}^C}(s_2, \theta, 4)$ are the SWNE values of the bimatrix game:

$$\mathsf{Z}_1 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} \frac{1}{3} & 2 \end{pmatrix} \end{array} \quad \text{and} \quad \mathsf{Z}_2 = \begin{array}{c c} & \begin{array}{cc} c & s \end{array} \\ \perp & \begin{pmatrix} 1 & \frac{1}{3} \end{pmatrix} \end{array}$$

i.e. the values $(\frac{1}{3}, 1)$.

As can be seen the values computed during value iteration oscillate for both s_1 and s_2.