Multi-player Equilibria Verification for Concurrent Stochastic Games

Marta Kwiatkowska[1] [0000−0001−9022−7599], Gethin Norman[2] [0000−0001−9326−4344], David Parker[3] [0000−0003−4137−8862], and Gabriel Santos[1] [0000−0002−6570−9737]

[1] Department of Computing Science, University of Oxford, UK
[2] School of Computing Science, University of Glasgow, UK
[3] School of Computer Science, University of Birmingham, UK

Abstract. Concurrent stochastic games (CSGs) are an ideal formalism for modelling probabilistic systems that feature multiple players or components with distinct objectives making concurrent, rational decisions. Examples include communication or security protocols and multi-robot navigation. Verification methods for CSGs exist but are limited to scenarios where agents or players are grouped into two *coalitions*, with those in the same coalition sharing an identical objective. In this paper, we propose *multi-coalitional* verification techniques for CSGs. We use subgame-perfect social welfare (or social cost) optimal Nash equilibria, which are strategies where there is no incentive for any coalition to unilaterally change its strategy in any game state, and where the total combined objectives are maximised (or minimised). We present an extension of the temporal logic rPATL (probabilistic alternating-time temporal logic with rewards) to specify equilibria-based properties for any number of distinct coalitions, and a corresponding model checking algorithm for a variant of stopping games. We implement our techniques in the PRISM-games tool and apply them to several case studies, including a secret sharing protocol and a public good game.

1 Introduction

Stochastic multi-player games are a modelling formalism that involves a number of players making sequences of rational decisions, each of which results in a probabilistic change in state. They are well suited to modelling systems that feature competitive or collaborative behaviour between multiple components or agents, operating in uncertain or stochastic environments. Examples include communication or security protocols, which may employ randomisation or send messages over unreliable channels, and multi-robot or multi-vehicle navigation, where sensors and actuators are subject to noise or prone to failure. A game-theoretic approach to modelling also allows rewards, incentives or resource usage to be incorporated. For example, mechanism design can be used to create protocols reliant on incentive schemes to improve robustness against selfish behaviour by participants, as utilised in network routing protocols [34] and auctions [14].

Designing reliable systems that comprise multiple components with differing objectives is a challenge. This is further complicated by the need to consider stochastic behaviour. Formal verification techniques for stochastic multi-player games can be a valuable tool for tackling this problem. The probabilistic model checker PRISM-games [25] has been developed for modelling and analysis of stochastic games: both the turn-based variant, where one player makes a decision in each state, and the concurrent variant, where players make decisions concurrently and without knowledge of each other's actions. PRISM-games also supports strategy synthesis, which allows automated generation of strategies for one or more players in the game, which are guaranteed to satisfy quantitative correctness specifications written in temporal logic.

The temporal logics used in PRISM-games for stochastic games are based on rPATL (probabilistic alternating-time temporal logic with rewards) [12], which combines features of the game logic ATL [4] and probabilistic temporal logics. For example, in a 3-player game, the formula $\langle\!\langle \mathtt{rbt_1} \rangle\!\rangle \mathtt{P}_{\geqslant p}[\,\mathtt{F}\ \mathtt{g_1}\,]$ states "robot 1 has a strategy under which the probability of it successfully reaching its goal is at least p, regardless of the strategies of robots 2 and 3". Model checking and strategy synthesis algorithms for rPATL exist for both turn-based [12] and concurrent stochastic games [22].

rPATL uses ATL's coalition operator $\langle\!\langle \cdot \rangle\!\rangle$ to formulate properties. In the above example, there are two coalitions, one containing robot 1 and the other robots 2 and 3. The coalitions have distinctly opposing (zero-sum) objectives, aiming either to maximise or minimise the probability of robot 1 reaching its goal. A recent extension [23] allows the two coalitions to have distinct objectives, using Nash equilibria. More precisely, it uses subgame-perfect social welfare optimal Nash equilibria, which are strategies for all players where there is no incentive for either coalition to unilaterally change its strategy in any state, and where the total combined objectives are maximised. For example, $\langle\!\langle \mathtt{rbt_1}{:}\mathtt{rbt_2}, \mathtt{rbt_3} \rangle\!\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\ \mathtt{g}_1\,] + \mathtt{P}[\,\mathtt{F}\ (\mathtt{g}_2 \wedge \mathtt{g}_3)\,])$ asks for such an equilibrium, where the two coalitions' objectives are to maximise the probability of reaching their own (distinct) goals. Model checking rPATL for both the zero-sum [12,22] and equilibria-based [23] properties has the advantage that it essentially reduces to the analysis of 2-player stochastic games, for which various algorithms exist (e.g., [2,3,10]). However, a clear limitation is the assumption that agents can, or would be willing to, collaborate and form two distinct coalitions.

In this paper, we propose *multi-coalitional* verification techniques for concurrent stochastic games (CSGs). We extend the temporal logic rPATL to allow reasoning about any number of distinct coalitions with different quantitative objectives, expressed using a variety of temporal operators capturing either the probability of an event occurring or a reward measure. We then give a model checking algorithm for the logic against CSGs, restricting our attention to a variant of stopping games [13], which, with probability 1, eventually reach a point where the outcome of each player's objective does not change by continuing. Our algorithm uses a combination of backward induction (for finite-horizon operators) and value iteration (for infinite-horizon operators). A key ingredient of

the computation is finding optimal Nash equilibria for n-player games, which we perform using support enumeration [33] and a mixture of SMT and non-linear optimisation solvers. We implement our techniques in the PRISM-games tool and apply them to several case studies, including a secret sharing protocol and a public good game. This allows us to verify multi-player scenarios that could not be analysed with existing techniques [23].

Related work. As summarised above, there are various algorithms to solve CSGs, e.g., [2,3,10], and model checking techniques have been developed for both zero-sum [22] and equilibria-based [23] versions of rPATL on CSGs, implemented in PRISM-games [25]. However, all of this work assumes or reduces to the 2-player case. Equilibria for n-player CSGs are considered in [8], but only complexity results, not algorithms, are presented. Other tools exist to reason about equilibria, including PRALINE [7], EAGLE [37], EVE [18], MCMAS-SLK [9] (via strategy logic) and Gambit [26], but these are all for *non-stochastic* games.

2 Preliminaries

We let $Dist(X)$ denote the set of probability distributions over set X. For any vector v we use $v(i)$ to denote the ith entry of the vector.

Definition 1 (Normal form game). *A (finite, n-person)* normal form game *(NFG) is a tuple $\mathsf{N} = (N, A, u)$ where: $N = \{1, \ldots, n\}$ 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 $i \in N$; $u = (u_1, \ldots, u_n)$ and $u_i : A \to \mathbb{R}$ is a utility function for player $i \in N$.*

For an NFG N, the players choose actions at the same time, where the choice for player $i \in N$ is over the action set A_i. When each player i chooses a_i, the utility received by player j equals $u_j(a_1, \ldots, a_n)$. A (mixed) strategy σ_i for player i is a distribution over its action set. Let η_{a_i} denote the pure strategy that selects action a_i with probability 1 and Σ^i_{N} the set of strategies for player i. A *strategy profile* $\sigma = (\sigma_1, \ldots, \sigma_n)$ is a tuple of strategies for each player and under σ the expected utility of player i equals:

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

For strategy σ_i of a player, the *support* of σ_i is the set of actions it chooses with nonzero probability, i.e., $\{a_i \in A_i \mid \sigma_i(a_i) > 0\}$. Furthermore, the support of a profile is the product of the supports of the individual strategies and a profile is said to have full support if it includes all available action tuples.

We now fix an NFG $\mathsf{N} = (N, A, u)$ and introduce the notion of Nash equilibrium and the variants we require. For profile $\sigma = (\sigma_1, \ldots, \sigma_n)$ and player i strategy σ'_i, we define the sequence $\sigma_{-i} \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_n)$ and profile $\sigma_{-i}[\sigma'_i] \stackrel{\text{def}}{=} (\sigma_1, \ldots, \sigma_{i-1}, \sigma'_i, \sigma_{i+1}, \ldots, \sigma_n)$.

Definition 2 (Best and least response). *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*

$u_i(\sigma_{-i}[\sigma_i^\star]) \geqslant u_i(\sigma_{-i}[\sigma_i])$ *for all* $\sigma_i \in \Sigma_{\mathsf{N}}^i$ *and a* least response *for player* i *to* σ_{-i} *is a strategy* $\sigma_i^\star$ *for player* i *such that* $u_i(\sigma_{-i}[\sigma_i^\star]) \leqslant u_i(\sigma_{-i}[\sigma_i])$ *for all* $\sigma_i \in \Sigma_{\mathsf{N}}^i$.

Definition 3 (Nash equilibrium). *A strategy profile* $\sigma^\star$ *is a* Nash equilibrium *(NE) if* $\sigma_i^\star$ *is a best response to* $\sigma_{-i}^\star$ *for all* $i \in N$.

Definition 4 (Social welfare NE). *An NE* $\sigma^\star$ *is a* social welfare optimal NE *(SWNE) and* $\langle u_i(\sigma^\star)\rangle_{i\in N}$ *are* 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 5 (Social cost NE). *A profile* $\sigma^\star$ *of* N *is a* social cost optimal NE *(SCNE) and* $\langle u_i(\sigma^\star)\rangle_{i\in N}$ *are* SCNE values *if* $\sigma^\star$ *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* N^-. *Furthermore,* $\sigma^\star$ *is an SWNE of* N^- *if and only if* $\sigma^\star$ *is an SCNE of* N.

The notion of SWNE is standard [29] and applies when utility values represent profits or rewards. We use the dual notion of SCNE for utilities that represent losses or costs. Example objectives in this category include minimising the probability of a fault or the expected time to complete a task. We have chosen to represent SCNE directly since this is more natural than the alternative of simply negating utilities, particularly in the case of probabilities.

Example 1. Consider the NFG representing a variant of a *public good* game [20], in which three players each receive a fixed amount of capital (10€) and can choose to invest none, half or all of it in a common stock (represented by the actions in_i^0, in_i^5 and in_i^{10} respectively). The total invested by the players is multiplied by a factor f and distributed equally among the players, and the aim of the players is to maximise their profit. The utility function of player i is therefore given by:

$$u_i(in_1^{k_1}, in_2^{k_2}, in_3^{k_3}) = (f/3)\cdot(k_1{+}k_2{+}k_3) - k_i\,.$$

for $k_i \in \{0, 5, 10\}$ and $1{\leqslant}i{\leqslant}3$. If $f{=}2$, then the profile where each investor chooses not to invest is an NE and each player's utility equals 0. More precisely, if a single player was to deviate from this profile by investing half or all of their capital, then their utility would decrease to $(2/3)\cdot5{-}5 = -5/3$ or $(2/3)\cdot10{-}10 = -10/3$, respectively. Since this is the only NE it is also the only SWNE and $(0, 0, 0)$ are the only SWNE values. The profile where each player invests all of their capital is not an NE as, under this profile, a player's utility equals $(2/3)\cdot30{-}10 = 10$ and any player can increase their utility to $(2/3)\cdot25{-}5 = 35/3$ by deviating and investing half of their capital.

On the other hand, if $f{=}3$, then there are two NE: when all players invest either none or invest all of their capital. The sum of utilities of the players under these profiles are $0{+}0{+}0 = 0$ and $20{+}20{+}20 = 60$ respectively, and therefore the second profile is the only SWNE.

Definition 6 (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:*

- $N = \{1, \ldots, n\}$ *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 (partial) 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 $A_i(s) \stackrel{\text{def}}{=} \Delta(s) \cap A_i$ if this set is non-empty, and from $\{\bot\}$ otherwise. For any state s and action tuple $a = (a_1, \ldots, a_n)$, the partial probabilistic transition function δ is defined for (s, a) if and only if $a_i \in A_i(s)$ for all $i \in N$. We augment CSGs with *reward structures*, which are tuples of the form $r{=}(r_A, r_S)$ where $r_A : S{\times}A \to \mathbb{R}$ and $r_S : S \to \mathbb{R}$ are action and state reward functions, respectively.

A *path* is a sequence $\pi = s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} \cdots$ such that $s_i \in S$, $\alpha_i = (a_1^i, \ldots, a_n^i) \in A$, $a_j^i \in A_j(s_i)$ for $j \in N$ and $\delta(s_i, \alpha_i)(s_{i+1}){>}0$ for all $i{\geqslant}0$. Given a path π, we denote by $\pi(i)$ the $(i{+}1)$th state, $\pi[i]$ the $(i{+}1)$th action, and if π is finite, $\mathit{last}(\pi)$ the final state. The sets of finite and infinite paths (starting in state s) of G are given by $\mathit{FPaths}_{\mathsf{G}}$ and $\mathit{IPaths}_{\mathsf{G}}$ ($\mathit{FPaths}_{\mathsf{G},s}$ and $\mathit{IPaths}_{\mathsf{G},s}$).

Strategies are used to resolve the choices of the players. Formally, a strategy for player i is a function $\sigma_i \colon \mathit{FPaths}_{\mathsf{G}} \to \mathit{Dist}(A_i \cup \{\bot\})$ such that, if $\sigma_i(\pi)(a_i){>}0$, then $a_i \in A_i(\mathit{last}(\pi))$. A *strategy profile* is a tuple $\sigma{=}(\sigma_1, \ldots, \sigma_n)$ of strategies for all players. The set of strategies for player i and set of profiles are denoted Σ_{G}^i and Σ_{G}. Given a profile σ and state s, let $\mathit{IPaths}_{\mathsf{G},s}^{\sigma}$ denote the infinite paths with initial state s corresponding to σ. We can then define, using standard techniques [21], a probability measure $\mathit{Prob}_{\mathsf{G},s}^{\sigma}$ over $\mathit{IPaths}_{\mathsf{G},s}^{\sigma}$ and, for a random variable $X \colon \mathit{IPaths}_{\mathsf{G}} \to \mathbb{R}$, the expected value $\mathbb{E}_{\mathsf{G},s}^{\sigma}(X)$ of X in s under σ.

In a CSG, a player's utility or *objective* is represented by a random variable $X_i \colon \mathit{IPaths}_{\mathsf{G}} \to \mathbb{R}$. Such variables can encode, for example, the probability of reaching a target or the expected cumulative reward before reaching a target. Given an objective for each player, social welfare and social cost NE can be defined as for NFGs. As in [23], we consider *subgame-perfect* NE [32], which are NE in *every state* of the CSG. In addition, for infinite-horizon objectives, the existence of NE is an open problem [6] so, for such objectives, we use ε-NE, which exist for any $\varepsilon{>}0$. Formally, we have the following definition.

Definition 7 (Subgame-perfect ε-NE). *For CSG* G *and* $\varepsilon{>}0$*, a profile* $\sigma^\star$ *is a* subgame-perfect ε-NE *for the objectives* $\langle X_i \rangle_{i \in N}$ *if and only if:* $\mathbb{E}_{\mathsf{G},s}^{\sigma^\star}(X_i) \geqslant \sup_{\sigma_i \in \Sigma_i} \mathbb{E}_{\mathsf{G},s}^{\sigma^\star_{-i}[\sigma_i]}(X_i) - \varepsilon$ *for all* $i \in N$ *and* $s \in S$*.*

Example 2. We now extend Example 1 to allow the players to invest their capital (and subsequent profits) over a number of months and assume that, at the end of each month, the parameter f can either increase or decrease by 0.2 with probability 0.1. This can be modelled as a CSG G whose states are tuples of the form (m, f, c_1, c_2, c_3), where m is the current month, f the parameter value and c_i is the current capital of player i (the initial capital plus or minus any

profits or losses made in previous months). If f has initial value 2 and the players start with a capital of 10€, then the initial state of G equals $(0, 2, 10, 10, 10)$. The actions of player i are of the form $in_i^{k_i}$, which corresponds to i investing k_i in the current month. The probabilistic transition function of the game is such that:

$$\delta((m, f, c_1, c_2, c_3), (in_1^{k_1}, in_2^{k_2}, in_3^{k_3}))(m', f', c_1', c_2', c_3') = \begin{cases} 0.8 & \text{if } m'{=}m{+}1,\ f'{=}f \text{ and } c_i'{=}c_i{+}p_i \\ 0.1 & \text{if } m'{=}m{+}1,\ f'{=}f{+}0.2 \text{ and } c_i'{=}c_i{+}p_i \\ 0.1 & \text{if } m'{=}m{+}1,\ f'{=}f{-}0.2 \text{ and } c_i'{=}c_i{+}p_i \\ 0 & \text{otherwise} \end{cases}$$

where $p_i = (f/3){\cdot}(k_1{+}k_2{+}k_3) - k_i$ for $k_i \in \{0, 5, 10\}$ and $1{\leqslant}i{\leqslant}3$.

If we are interested in the profits of the players after k months, then we can consider a random variable for player i which would return, for a path with $(k{+}1)$th state (k, f, c_1, c_2, c_3), the value $c_i{-}10$.

3 Extended rPATL with Nash Formulae

We now consider the logic rPATL with Nash formulae [23] and enhance it with equilibria-based properties that can separate players into more than two coalitions.

Definition 8 (Extended rPATL syntax). *The syntax of our extended version of* rPATL *is given by the grammar:*

$$\begin{aligned} \phi &:= \mathtt{true} \mid \mathtt{a} \mid \neg\phi \mid \phi \wedge \phi \mid \langle\!\langle C \rangle\!\rangle \mathtt{P}_{\sim q}[\,\psi\,] \mid \langle\!\langle C \rangle\!\rangle \mathtt{R}^r_{\sim x}[\,\rho\,] \mid \langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\mathrm{opt}\sim x}(\theta) \\ \theta &:= \mathtt{P}[\,\psi\,]{+}\cdots{+}\mathtt{P}[\,\psi\,] \ \mid\ \mathtt{R}^r[\,\rho\,]{+}\cdots{+}\mathtt{R}^r[\,\rho\,] \\ \psi &:= \mathtt{X}\,\phi \ \mid\ \phi\ \mathtt{U}^{\leqslant k}\ \phi \ \mid\ \phi\ \mathtt{U}\ \phi \\ \rho &:= \mathtt{I}^{=k} \ \mid\ \mathtt{C}^{\leqslant k} \ \mid\ \mathtt{F}\ \phi \end{aligned}$$

where $\mathtt{a}$ *is an atomic proposition,* C *and* $C_1, \ldots, C_m$ *are coalitions of players such that* $C_i \cap C_j = \varnothing$ *for all* $1 \leqslant i \neq j \leqslant m$ *and* $\cup_{i=1}^m C_i = N$, $\mathrm{opt} \in \{\min, \max\}$, $\sim \in \{<, \leqslant, \geqslant, >\}$, $q \in \mathbb{Q} \cap [0, 1]$, $x \in \mathbb{Q}$, r *is a reward structure and* $k \in \mathbb{N}$.

Our addition to the logic is *Nash formulae* of the form $\langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\mathrm{opt}\sim x}(\theta)$, where the *nonzero sum* formulae θ comprises a sum of m probability or reward objectives (for full details of the rest of the logic see [22,23]). The formula $\langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\max\sim x}(\mathtt{P}[\,\psi_1\,]{+}\cdots{+}\mathtt{P}[\,\psi_m\,])$ holds in a state if, when the players form the coalitions $C_1, \ldots, C_m$, there is a subgame-perfect SWNE for which the *sum* of the values of the objectives $\mathtt{P}[\,\psi_1\,], \ldots, \mathtt{P}[\,\psi_m\,]$ for the coalitions $C_1, \ldots, C_m$ satisfies $\sim x$. The case for reward objectives is similar and, for formulae of the form $\langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\min\sim x}(\theta)$, we require the existence of an SCNE rather than an SWNE. We also allow *numerical* queries of the form $\langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\mathrm{opt}=?}(\theta)$, which return the sum of the SWNE or SCNE values.

In a probabilistic nonzero-sum formula $\theta{=}\mathtt{P}[\,\psi_1\,]{+}\cdots{+}\mathtt{P}[\,\psi_m\,]$, each objective ψ_i can be a next ($\mathtt{X}\,\phi$), bounded until ($\phi_1\ \mathtt{U}^{\leqslant k}\ \phi_2$) or until ($\phi_1\ \mathtt{U}\ \phi_2$) formula, with the usual equivalences, e.g., $\mathtt{F}\ \phi \equiv \mathtt{true}\ \mathtt{U}\ \phi$. For the reward case

$\theta{=}\mathtt{R}^{r_1}[\,\rho_1\,]{+}\cdots{+}\mathtt{R}^{r_m}[\,\rho_m\,]$, each ρ_i refers to a reward formula with respect to reward structure r_i and can be bounded instantaneous reward ($\mathtt{I}^{=k}$), bounded accumulated reward ($\mathtt{C}^{\leqslant k}$) or reachability reward ($\mathtt{F}\ \phi$).

Example 3. Recall the public good CSG from Example 2. Examples of nonzero-sum formulae in our logic include:

- $\langle\!\langle p_1{:}p_2{:}p_3\rangle\!\rangle_{\max\geqslant 3}(\mathtt{P}[\,\mathtt{F}\ c_1{\geqslant}20\,]{+}\mathtt{P}[\,\mathtt{F}\ c_2{\geqslant}20\,]{+}\mathtt{P}[\,\mathtt{F}\ c_3{\geqslant}20\,])$ states that the three players can collaborate such that they each eventually double their capital with probability 1;
- $\langle\!\langle p_1{:}p_2{:}p_3\rangle\!\rangle_{\max=?}(\mathtt{R}^{cap_1}[\,\mathtt{I}^{=4}\,]{+}\mathtt{R}^{cap_2}[\,\mathtt{I}^{=4}\,]{+}\mathtt{R}^{cap_3}[\,\mathtt{I}^{=4}\,])$ asks for the sum of the expected capital of the players at 4 months when they collaborate, where the state reward function of cap_i returns the capital of player i.
- $\langle\!\langle p_1{:}p_2{:}p_3\rangle\!\rangle_{\max\geqslant 50}(\mathtt{R}^{pro_1}[\,\mathtt{C}^{\leqslant 6}\,]{+}\mathtt{R}^{pro_2}[\,\mathtt{C}^{\leqslant 6}\,]{+}\mathtt{R}^{pro_3}[\,\mathtt{C}^{\leqslant 6}\,])$ states that the sum of the expected cumulative profit of the players after 6 months when they collaborate is at least 50, where the action reward function of pro_i returns the expected profit of player i from a state for the given action tuple.

In order to give the semantics of the logic, we require an extension of the notion of *coalition games* [22] which, given a CSG G and partition $\mathcal{C}$ of the players into m coalitions, reduces G to an m-player coalition game, where each player corresponds to one of the coalitions in $\mathcal{C}$. Without loss of generality, we assume $\mathcal{C}$ is of the form $\{\{1,\dots,n_1\},\{n_1{+}1,\dots n_2\},\dots,\{n_{m-1}{+}1,\dots n_m\}\}$ and let $j_{\mathcal{C}}$ denote player j's position in its coalition.

Definition 9 (Coalition game). *For CSG* $\mathsf{G}{=}(N,S,\bar{s},A,\Delta,\delta,AP,L)$ *and partition of the players into* m *coalitions* $\mathcal{C}=\{C_1,\dots,C_m\}$*, we define the* coalition game $\mathsf{G}^{\mathcal{C}}{=}(M,S,\bar{s},A^{\mathcal{C}},\Delta^{\mathcal{C}},\delta^{\mathcal{C}},AP,L)$ *as an* m*-player CSG where:*

- $M=\{1,\dots,m\}$*;*
- $A^{\mathcal{C}}=(A_1^{\mathcal{C}}\cup\{\bot\})\times\cdots\times(A_m^{\mathcal{C}}\cup\{\bot\})$*;*
- $A_i^{\mathcal{C}}=(\prod_{j\in C_i}(A_j\cup\{\bot\})\setminus\{(\bot,\dots,\bot)\})$ *for all* $i\in M$*;*
- *for any* $s\in S$ *and* $i\in M$*:* $a_i^{\mathcal{C}}\in\Delta^{\mathcal{C}}(s)$ *if and only if either* $\Delta(s)\cap A_j=\varnothing$ *and* $a_i^{\mathcal{C}}(j_{\mathcal{C}})=\bot$ *or* $a_i^{\mathcal{C}}(j_{\mathcal{C}})\in\Delta(s)$ *for all* $j\in C_i$*;*
- *for any* $s\in S$ *and* $(a_1^{\mathcal{C}},\dots,a_m^{\mathcal{C}})\in A^{\mathcal{C}}$*:* $\delta^{\mathcal{C}}(s,(a_1^{\mathcal{C}},\dots,a_m^{\mathcal{C}}))=\delta(s,(a_1,\dots,a_n))$ *where for* $i\in M$ *and* $j\in C_i$ *if* $a_i^{\mathcal{C}}{=}\bot$*, then* $a_j{=}\bot$ *and otherwise* $a_j{=}a_i^{\mathcal{C}}(j_{\mathcal{C}})$*.*

Furthermore, for a reward structure $r=(r_A,r_S)$*, by abuse of notation we use* $r=(r_A^{\mathcal{C}},r_S^{\mathcal{C}})$ *for the corresponding reward structure of* $\mathsf{G}^{\mathcal{C}}$ *where:*

- *for any* $s\in S$*,* $a_i^C\in A_i^C$*:* $r_{A^{\mathcal{C}}}^{\mathcal{C}}(s,(a_1^C,\dots,a_m^C))=r_A(s,(a_1,\dots,a_n))$ *where for* $i\in M$ *and* $j\in C_i$*, if* $a_i^{\mathcal{C}}=\bot$*, then* $a_j{=}\bot$ *and otherwise* $a_j{=}a_i^{\mathcal{C}}(j_{\mathcal{C}})$*;*
- *for any* $s\in S$*:* $r_S^{\mathcal{C}}(s){=}r_S(s)$*.*

The logic includes infinite-horizon objectives ($\mathtt{U}$, $\mathtt{F}$), for which the existence of SWNE and SCNE is open [6]. However, ε-SWNE and ε-SCNE *do* exist for any $\varepsilon>0$.

Definition 10 (Extended rPATL semantics). *For a CSG* G*,* $\varepsilon > 0$ *and a formula* ϕ*, the satisfaction relation* $\models$ *is defined inductively over the structure of* ϕ*. The propositional logic fragment* ($\mathtt{true}$, $\mathtt{a}$, $\neg$, $\wedge$) *is defined in the usual way. The zero-sum formulae* $\langle\!\langle C \rangle\!\rangle \mathtt{P}_{\sim q}[\,\psi\,]$ *and* $\langle\!\langle C \rangle\!\rangle \mathtt{R}^{r}_{\sim x}[\,\rho\,]$ *are defined as in* [22,23]*. For a Nash formula and state* $s \in S$ *in CSG* G*, we have:*

$$s \models \langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\mathrm{opt}\sim x}(\theta) \Leftrightarrow \exists \sigma^\star \in \Sigma_{\mathsf{G}^\mathcal{C}}.\ \left(\mathbb{E}^{\sigma^\star}_{\mathsf{G}^\mathcal{C},s}(X^\theta_1) + \cdots + \mathbb{E}^{\sigma^\star}_{\mathsf{G}^\mathcal{C},s}(X^\theta_m)\right) \sim x$$

and $\sigma^\star = (\sigma^\star_1, \ldots, \sigma^\star_m)$ *is a subgame perfect* ε*-SWNE if* $\mathrm{opt} = \max$*, and a subgame perfect* ε*-SCNE if* $\mathrm{opt} = \min$*, for the objectives* $(X^\theta_1, \ldots, X^\theta_m)$ *in* $\mathsf{G}^\mathcal{C}$ *where* $\mathcal{C} = \{C_1, \ldots, C_m\}$ *and for* $1 \leqslant i \leqslant m$ *and* $\pi \in \mathit{IPaths}^{\sigma^\star}_{\mathsf{G}^\mathcal{C},s}$ *:*

$$\begin{aligned}
X_i^{\mathtt{P}[\,\psi^1\,]+\cdots+\mathtt{P}[\,\psi^m\,]}(\pi) &= 1 \text{ if } \pi \models \psi^i \text{ and } 0 \text{ otherwise} \\
X_i^{\mathtt{R}^{r_1}[\,\rho^1\,]+\cdots+\mathtt{R}^{r_m}[\,\rho^m\,]}(\pi) &= \mathit{rew}(r_i, \rho^i)(\pi) \\
\pi \models \mathtt{X}\,\phi &\Leftrightarrow \pi(1) \models \phi \\
\pi \models \phi_1\ \mathtt{U}^{\leqslant k}\ \phi_2 &\Leftrightarrow \exists i \leqslant k.\, (\pi(i) \models \phi_2 \wedge \forall j < i.\, \pi(j) \models \phi_1) \\
\pi \models \phi_1\ \mathtt{U}\ \phi_2 &\Leftrightarrow \exists i \in \mathbb{N}.\, (\pi(i) \models \phi_2 \wedge \forall j < i.\, \pi(j) \models \phi_1) \\
\mathit{rew}(r, \mathtt{I}^{=k})(\pi) &= r_S(\pi(k)) \\
\mathit{rew}(r, \mathtt{C}^{\leqslant k})(\pi) &= \textstyle\sum_{i=0}^{k-1} \big(r_A(\pi(i), \pi[i]) + r_S(\pi(i))\big) \\
\mathit{rew}(r, \mathtt{F}\ \phi)(\pi) &= \begin{cases} \infty & \text{if } \forall j \in \mathbb{N}.\, \pi(j) \not\models \phi \\ \sum_{i=0}^{k_\phi} \big(r_A(\pi(i), \pi[i]) + r_S(\pi(i))\big) & \text{otherwise} \end{cases}
\end{aligned}$$

and $k_\phi = \min\{k{-}1 \mid \pi(k) \models \phi\}$.

4 Model Checking CSGs against Nash Formulae

rPATL is a branching-time logic and so the model checking algorithm works by recursively computing the set $\mathit{Sat}(\phi)$ of states satisfying formula ϕ over the structure of ϕ. Therefore, to extend the existing algorithm of [22,23], we need only consider formulae of the form $\langle\!\langle C_1 {:}\cdots{:} C_m \rangle\!\rangle_{\mathrm{opt}\sim x}(\theta)$. From Definition 10, this requires the computation of subgame-perfect SWNE or SCNE values of the objectives $(X^\theta_1, \ldots, X^\theta_m)$ and a comparison of their sum to the threshold x.

We first explain how we compute SWNE values in NFGs. Next we consider CSGs, and show how to compute subgame-perfect SWNE and SCNE values for finite-horizon objectives and approximate values for infinite-horizon objectives. For the remainder of this section we fix an NFG N and CSG G.

As in [23], to check nonzero-sum properties on CSGs, we have to work with a restricted class of games. This can be seen as a variant of *stopping games* [13], as used for multi-objective turn-based stochastic games. Compared to [13], 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. This can be checked using graph algorithms [1].

Assumption 1. *For each subformula* $\mathtt{P}[\,\phi_1^i\ \mathtt{U}\ \phi_2^i\,]$*, set* $Sat(\neg\phi_1^i \vee \phi_2^i)$ *is reached with probability 1 from all states under all profiles. For each subformula* $\mathtt{R}^r[\,\mathtt{F}\ \phi^i\,]$*, the set* $Sat(\phi^i)$ *is reached with probability 1 from all states under all profiles.*

Computing SWNE Values of NFGs. Computing NE values for an n-player game is a complex task when $n{>}2$, as it can no longer be reduced to a linear programming problem. The algorithm for the two-player case presented in [23], based on *labelled polytopes*, starts by considering all the regions of the strategy profile space and then iteratively reduces the search space as positive probability assignments are found and added as restrictions on this space. The efficiency of this approach deteriorates when analysing games with large numbers of actions and when one or more players are indifferent, as the possible assignments resulting from action permutations need to be exhausted.

Going in the opposite direction, support enumeration [33] is a method for computing NE that exhaustively examines all sub-regions, i.e., supports, of the strategy profile space, one at a time, checking whether that sub-region contains equilibria. The number of supports is exponential in the number of actions and equals $\prod_{i=1}^{n}(2^{|A_i|}-1)$. Therefore computing SWNE values through support enumeration will only be efficient for games with a small number of actions.

We now show how, for a given support, using the following lemma, the computation of SWNE profiles can be encoded as a *nonlinear programming problem*. The lemma states that a profile is an NE if and only if any player switching to a single action in the support of the profile yields the same utility for the player and switching to an action outside the support can only decrease its utility.

Lemma 1 ([33]). *The strategy profile* $\sigma{=}(\sigma_1,\ldots,\sigma_n)$ *of* N *is an NE if and only if the following conditions are satisfied:*

$$\forall i \in N.\, \forall a_i \in A_i.\ \ \sigma_i(a_i) > 0 \rightarrow u_i(\sigma_{-i}[\eta_{a_i}]) = u_i(\sigma) \tag{1}$$

$$\forall i \in N.\, \forall a_i \in A_i.\ \ \sigma_i(a_i) = 0 \rightarrow u_i(\sigma_{-i}[\eta_{a_i}]) \leqslant u_i(\sigma)\,. \tag{2}$$

Given the support $B = B_1{\times}\cdots{\times}B_n \subseteq A$, to construct the problem, we first choose *pivot* actions[4] $b_i^p \in B_i$ for $i \in N$, then the problem is to minimise:

$$\left(\textstyle\sum_{i\in N} \max_{a\in A} u_i(a)\right) - \sum_{i\in N}\left(\sum_{b\in B} u_i(b)\cdot\left(\prod_{j\in N} p_{j,b_j}\right)\right) \tag{3}$$

subject to:

$$\textstyle\sum_{c\in B_{-i}(b_i^p)} u_i(c)\cdot\left(\prod_{j\in N_{-i}} p_{j,c_j}\right) - \sum_{c\in B_{-i}(b_i)} u_i(c)\cdot\left(\prod_{j\in N_{-i}} p_{j,c_j}\right) = 0 \tag{4}$$

$$\textstyle\sum_{c\in B_{-i}(b_i^p)} u_i(c)\cdot\left(\prod_{j\in N_{-i}} p_{j,c_j}\right) - \sum_{c\in B_{-i}(a_i)} u_i(c)\cdot\left(\prod_{j\in N_{-i}} p_{j,c_j}\right) \geqslant 0 \tag{5}$$

$$\textstyle\sum_{b_i\in B_i} p_{i,b_i} = 1 \quad\text{and}\quad p_{i,b_i} > 0 \tag{6}$$

for all $i \in N$, $b_i \in B_i\backslash\{b_i^p\}$ and $a_i \in A_i\backslash B_i$ where $B_{-i}(c_i) = B_1{\times}\cdots{\times}B_{i-1}{\times}\{c_i\}{\times}B_{i+1}{\times}\cdots{\times}B_n$ and $N_{-i} = N\backslash\{i\}$. The variables in the above program represent the probabilities players choose different actions, i.e. p_{i,b_i} is the probability

[4] For each $i \in N$ this can be any action in B_i.

a	u_1	u_2	u_3
(c_1, c_2, c_3)	7	7	7
(c_1, c_2, d_3)	3	3	9

a	u_1	u_2	u_3
(c_1, d_2, c_3)	3	9	3
(c_1, d_2, d_3)	0	5	5

a	u_1	u_2	u_3
(d_1, c_2, c_3)	9	3	3
(d_1, c_2, d_3)	5	0	5

a	u_1	u_2	u_3
(d_1, d_2, c_3)	5	5	0
(d_1, d_2, d_3)	1	1	1

Table 1: Utilities for an instance of a three-player prisoner's dilemma.

i selects b_i. The constraints (6) ensure the probabilities of each player sum to one and the support of the corresponding profile equals B. The constraints (4) and (5) require that the solution corresponds to an NE as these encode the constraints (1) and (2), respectively, of Lemma 1 when restricting to pivot actions. This restriction is sufficient as (1) requires all actions in the support to yield the same utility. The first term in (3) corresponds to the maximum possible sum of utilities for the players, i.e. it sums the maximum utility of each player, and the second sums the individual utilities of the players when they play according to the profile corresponding to the solution. By minimising the difference between these two terms, we require the solution to be social welfare optimal.

SMT solvers with nonlinear modules can be used to solve such problems, although they can be inefficient. Alternative approaches include *barrier* or *interior-point* methods [30].

Example 4. Consider the instance of three prisoner's dilemma with utilities described in Table 1 where $A_i{=}\{c_i, d_i\}$ for $1{\leqslant}i{\leqslant}3$. For the full support B^{fs} the utility of player i equals:

$$u_i(B^{fs}) = p_{i,c_i}{\cdot}u_i(B^{fs}_{-i}(c_i)) + p_{i,d_i}{\cdot}u_i(B^{fs}_{-i}(d_i))$$

where $u_i(B^{fs}_{-i}(c_i))$ and $u_i(B^{fs}_{-i}(d_i))$ are the utilities of player i when switching to choosing action c_i and d_i with probability 1 and are given by:

$$\begin{aligned} u_i(B^{fs}_{-i}(c_i)) &= 7{\cdot}p_{j,c_j}{\cdot}p_{k,c_k} + 3{\cdot}p_{j,c_j}{\cdot}p_{k,d_k} + 3{\cdot}p_{j,d_j}{\cdot}p_{k,c_k} \\ u_i(B^{fs}_{-i}(d_i)) &= 9{\cdot}p_{j,c_j}{\cdot}p_{k,c_k} + 5{\cdot}p_{j,c_j}{\cdot}p_{k,d_k} + 5{\cdot}p_{j,d_j}{\cdot}p_{k,c_k} + p_{j,d_j}{\cdot}p_{k,d_k} \end{aligned}$$

for $1{\leqslant}i{\neq}j{\neq}k{\leqslant}3$. Now, choosing c_i as the pivot action for $1{\leqslant}i{\leqslant}3$, we obtain the nonlinear program of minimising:

$$27 - (u_1(B^{fs}) + u_2(B^{fs}) + u_3(B^{fs}))$$

subject to: $u_i(B^{fs}_{-i}(c_i)){-}u_i(B^{fs}_{-1}(d_i)) = 0$, $p_{i,c_i}{+}p_{i,d_i} = 1$, $p_{i,c_i}{>}0$ and $p_{i,d_i}{>}0$ for $1{\leqslant}i{\leqslant}3$. When trying to solving this problem, we find that there is no NE as the constraints reduce to $p_{3,c_3}{\cdot}(p_{2,d_2} + 1) = -1$, which cannot be satisfied.

For the partial support $B^{ps}{=}\{(d_1, d_2, d_3)\}$, d_i is the only choice of pivot action for player i and, after a reduction, we obtain the program of minimising:

$$27 - (p_{2,d_2}p_{3,d_3} + p_{1,d_1}p_{3,d_3} + p_{1,d_1}p_{2,d_2})$$

subject to: $p_{i,d_i}{\cdot}p_{j,d_j} \geqslant 0$, $p_{i,d_i}{=}1$ and $p_{i,d_i}{>}0$ for $1{\leqslant}i{\neq}j{\leqslant}3$. Solving this problem we see it is satisfied, and therefore the profile where each player i chooses d_i is an NE. This demonstrates that, as for the two-player prisoner's dilemma, defection dominates cooperation for all players, which leads to the only NE.

Computing Values of Nash Formulae. We now show how to compute the SWNE values of a Nash formula $\langle\!\langle C_1 : \cdots : C_m \rangle\!\rangle_{\mathrm{opt}\sim x}(\theta)$. The case for SCNE values can be computed similarly, where to compute SCNE values of a NFG N, we use Definition 5, negate the utilities of N, find SWNE values of the resulting NFG and return the negation of these values as SCNE values of N.

If all the objectives in the nonzero sum formula θ are finite-horizon, *backward induction* [35,28] can be applied to compute (precise) subgame-perfect SWNE values. On the other hand, when all the objectives are infinite-horizon, we extend the techniques of [23] for two coalitions and use *value iteration* [11] to approximate subgame-perfect SWNE values. In cases when there is a combination of finite- and infinite-horizon objectives, we can extend the techniques of [23] and make all objectives infinite-horizon by modifying the game in a standard manner.

Value computation for each type of objective is described below. The extension of the two-player case [23] is non-trivial: in that case, when one player reaches their goal, we can apply MDP verification techniques by making the players form a coalition to reach the remaining goal. However, in the n-player case, if one player reaches their goal we cannot reduce the analysis to an $(n-1)$-player game, as the choices of the player that has reached its goal can still influence the outcomes of the remaining players, and making the player form a coalition with one of the other players will give the other player an advantage. Instead, we need to keep track of the set of players that have reached their goal (denoted D) and can no longer reach their goal in the case of until formulae (denoted E), and define the values at each iteration using these sets.

We use the notation $\mathtt{V}_{\mathsf{G}^c}(s,\theta)$ ($\mathtt{V}_{\mathsf{G}^c}(s,\theta,n)$) for the vector of computed values of the objectives $(X_1^\theta, X_2^\theta, \ldots, X_m^\theta)$ in state s of $\mathsf{G}^{\mathcal{C}}$ (at iteration n). We also use $\mathbf{1}_m$ and $\mathbf{0}_m$ to denote a vector of size m whose entries all equal to 1 or 0, respectively. For any set of states S' and state s we let $\eta_{S'}(s)$ equal 1 if $s \in S'$ and 0 otherwise. Furthermore, to simplify the presentation the step bounds appearing in path and reward formulae can take negative values.

Bounded Probabilistic Until. If $\theta = \mathtt{P}[\,\phi_1^1\ \mathtt{U}^{\leqslant k_1}\ \phi_2^1\,] + \cdots + \mathtt{P}[\,\phi_1^m\ \mathtt{U}^{\leqslant k_m}\ \phi_2^m\,]$, we compute SWNE values of the objectives for the nonzero-sum formulae $\theta_n = \mathtt{P}[\,\phi_1^1\ \mathtt{U}^{\leqslant k_1 - n}\ \phi_2^1\,] + \cdots + \mathtt{P}[\,\phi_1^m\ \mathtt{U}^{\leqslant k_m - n}\ \phi_2^m\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \max\{k_1, \ldots, k_l\}$ and $\mathtt{V}_{\mathsf{G}^c}(s,\theta) = \mathtt{V}_{\mathsf{G}^c}(s,\varnothing,\varnothing,\theta_0)$. For any state s and $0 \leqslant n \leqslant k$, $D, E \subseteq M$ such that $D \cap E = \varnothing$:

$$\mathtt{V}_{\mathsf{G}^c}(s,D,E,\theta_n) = \begin{cases} (\eta_D(1), \ldots, \eta_D(m)) & \text{if } D \cup E = M \\ \mathtt{V}_{\mathsf{G}^c}(s, D \cup D', E, \theta_n) & \text{else if } D' \neq \varnothing \\ \mathtt{V}_{\mathsf{G}^c}(s, D, E \cup E', \theta_n) & \text{else if } E' \neq \varnothing \\ val(\mathsf{N}) & \text{otherwise} \end{cases}$$

where $D' = \{l \in M \backslash (D \cup E) \mid s \in Sat(\phi_2^l)\}$, $E' = \{l \in M \backslash (D \cup E) \mid s \in Sat(\neg\phi_1^l \wedge \neg\phi_2^l)\}$ and $val(\mathsf{N})$ equals SWNE values of the game $\mathsf{N} = (M, A^{\mathcal{C}}, u)$

in which for any $1 \leqslant l \leqslant m$ and $a \in A^{\mathcal{C}}$:

$$u_l(a) = \begin{cases} 1 & \text{if } l \in D \\ 0 & \text{else if } l \in E \\ 0 & \text{else if } n_l - n \leqslant 0 \\ \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot v_{n-1}^{s',l} & \text{otherwise} \end{cases}$$

and $(v_{n-1}^{s',1}, v_{n-1}^{s',2}, \ldots, v_{n-1}^{s',m}) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s', D, E, \theta_{n-1})$ for all $s' \in S$.

Instantaneous Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{I}^{=k_1}\,] + \cdots + \mathtt{R}^{r_m}[\,\mathtt{I}^{=k_m}\,]$, we compute SWNE values of the objectives for the nonzero-sum formulae $\theta_n = \mathtt{R}^{r_1}[\,\mathtt{I}^{=n_1-n}\,] + \cdots + \mathtt{R}^{r_m}[\,\mathtt{I}^{=n_l-n}\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \max\{k_1, \ldots, k_l\}$ and $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta_0)$. For any state s and $0 \leqslant n \leqslant k$, $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta_n)$ equals SWNE values of the game $\mathsf{N} = (M, A^{\mathcal{C}}, u)$ in which for any $1 \leqslant l \leqslant m$ and $a \in A^{\mathcal{C}}$:

$$u_l(a) = \begin{cases} 0 & \text{if } n_l - n < 0 \\ \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot r_S^l(s') & \text{else if } n_l - n = 0 \\ \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot v_{n+1}^{s',l} & \text{otherwise} \end{cases}$$

and $(v_{n+1}^{s',1}, \ldots, v_{n+1}^{s',m}) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s', \theta_{n+1})$ for all $s' \in S$.

Bounded Cumulative Rewards. If $\theta = \mathtt{R}^{r_1}[\,\mathtt{C}^{\leqslant k_1}\,] + \cdots + \mathtt{R}^{r_m}[\,\mathtt{C}^{\leqslant k_m}\,]$, we compute SWNE values of the objectives for the nonzero-sum formulae $\theta_n = \mathtt{R}^{r_1}[\,\mathtt{C}^{\leqslant n_1-n}\,] + \cdots + \mathtt{R}^{r_l}[\,\mathtt{C}^{\leqslant n_m-n}\,]$ for $0 \leqslant n \leqslant k$ recursively, where $k = \max\{k_1, \ldots, k_l\}$ and $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta_0)$. For any state s and $0 \leqslant n \leqslant k$, $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta_n)$ equals SWNE values of the game $\mathsf{N} = (M, A^{\mathcal{C}}, u)$ in which for any $1 \leqslant l \leqslant m$ and $a \in A^{\mathcal{C}}$:

$$u_l(a) = \begin{cases} 0 & \text{if } n_l - n \leqslant 0 \\ r_S^l(s) + r_A^l(s,a) + \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot v_{n+1}^{s',l} & \text{otherwise} \end{cases}$$

and $(v_{n+1}^{s',1}, \ldots, v_{n+1}^{s',m}) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s', \theta_{n+1})$ for all $s' \in S$.

Probabilistic Until. If $\theta = \mathtt{P}[\,\phi_1^1\ \mathtt{U}\ \phi_2^1\,] + \cdots + \mathtt{P}[\,\phi_1^m\ \mathtt{U}\ \phi_2^m\,]$, values can be computed through value iteration as the limit $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta) = \lim_{n\to\infty} \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta,n)$ where $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta,n) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\varnothing,\varnothing,\theta,n)$ and for any $D, E \subseteq M$ such that $D \cap E = \varnothing$:

$$\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,D,E,\theta,n) = \begin{cases} (\eta_D(1), \ldots, \eta_D(m)) & \text{if } D \cup E = M \\ (\eta_{Sat(\phi_2^1)}(s), \ldots, \eta_{Sat(\phi_2^m)}(s)) & \text{else if } n = 0 \\ \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s, D \cup D', E, \theta, n) & \text{else if } D' \neq \varnothing \\ \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s, D, E \cup E', \theta, n) & \text{else if } E' \neq \varnothing \\ val(\mathsf{N}) & \text{otherwise} \end{cases}$$

where $D' = \{l \in M \backslash (D \cup E) \mid s \in Sat(\phi_2^l)\}$, $E' = \{l \in M \backslash (D \cup E) \mid s \in Sat(\neg\phi_1^l \wedge \neg\phi_2^l)\}$ and $val(\mathsf{N})$ equals SWNE values of the game $\mathsf{N} = (M, A^{\mathcal{C}}, u)$ in which for any $1 \leqslant l \leqslant m$ and $a \in A^{\mathcal{C}}$:

$$u_l(a) = \begin{cases} 1 & \text{if } l \in D \\ 0 & \text{else if } l \in E \\ \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot v_{n-1}^{s',l} & \text{otherwise} \end{cases}$$

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

Expected Reachability. If $\theta = \mathtt{R}^{r_1}[\mathtt{F}\ \phi^1] + \cdots + \mathtt{R}^{r_m}[\mathtt{F}\ \phi^m]$, values can be computed through value iteration as the limit $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta) = \lim_{n\to\infty} \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta,n)$ where $\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\theta,n) = \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,\varnothing,\theta,n)$ and for any $D \subseteq M$:

$$\mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s,D,\theta,n) = \begin{cases} \mathbf{0}_m & \text{if } D = M \\ \mathbf{0}_m & \text{else if } n = 0 \\ \mathtt{V}_{\mathsf{G}^{\mathcal{C}}}(s, D \cup D', \theta, n) & \text{else if } D' \neq \varnothing \\ val(\mathsf{N}) & \text{otherwise} \end{cases}$$

$D' = \{l \in M \backslash D \mid s \in Sat(\phi^l)\}$ and $val(\mathsf{N})$ equals SWNE values of the game $\mathsf{N} = (M, A^{\mathcal{C}}, u)$ in which for any $1 \leqslant l \leqslant m$ and $a \in A^{\mathcal{C}}$:

$$u_l(a) = \begin{cases} 0 & \text{if } l \in D \\ r_S^l(s) + r_A^l(s,a) + \sum_{s' \in S} \delta^{\mathcal{C}}(s,a)(s') \cdot v_{n-1}^{s',l} & \text{otherwise} \end{cases}$$

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

Strategy Synthesis. When performing property verification, it is usually beneficial to include *strategy synthesis*, that is, construct a witness to the satisfaction of a property. When verifying a Nash formula $\langle\!\langle C_1 : \cdots : C_m \rangle\!\rangle_{\mathrm{opt} \sim x}(\theta)$, we can also return a subgame-perfect SWNE or SCNE for the objectives $(X_1^\theta, \ldots, X_m^\theta)$. This is achieved by keeping track of an SWNE for the NFG solved in each state. The synthesised strategies require randomisation and memory. Randomisation is needed for NE of NFGs. Memory is required for finite-horizon properties and since choices change after a path formula becomes true or a target is reached. For infinite-horizon properties, only approximate ε-NE profiles are synthesised.

Correctness and Complexity. The proof of correctness of the algorithm can be found in an extended version of this paper [24]. In the case of finite-horizon nonzero-sum formulae the correctness of the model checking algorithm follows from the fact that we use backward induction [35,28]. For infinite-horizon nonzero-sum formulae the proof is based on showing that the values of the players computed during value iteration correspond to subgame-perfect SWNE or SCNE values of finite game trees, and the values of these game trees converge uniformly to the actual values of $\mathsf{G}^{\mathcal{C}}$. The complexity of the algorithm is linear in the formula size, and finding subgame-perfect NE for reachability objectives in n-player games is PSPACE [8]. Value iteration requires finding all NE for a NFG in each state of the model, and computing NE of an NFG with three (or more) players is PPAD-complete [15].

5 Case Studies and Experimental Results

We have implemented our approach on top of PRISM-games 3.0 [25], extending the implementation to support multi-coalitional equilibria-based properties. The files for the case studies and results in this section are available from [41].

Game	Players	Actions	Supports	Supports returned by Z3			Time(s)
				unsat	*sat*	*unknown*	IPOPT
Majority Voting	3	3,3,3	343	330	12	1	0.309
	3	4,4,4	3,375	3,236	110	29	18.89
	3	5,5,5	29,791	26,250	155	3,386	336.5
	4	2,2,2,2	81	59	22	0	0.184
	4	3,3,3,3	2,401	2,212	87	102	6.847
	4	4,4,4,4	50,625	41,146	518	8,961	1,158
	5	2,2,2,2,2	243	181	62	0	0.591
	5	3,3,3,3,3	16,807	14,950	266	1,591	253.3
Covariant game	3	3,3,3	343	304	6	33	7.645
	3	4,4,4	3,375	2,488	16	871	203.8
	3	5,5,5	29,791	14,271	8	15,512	5,801
	4	2,2,2,2	81	76	3	2	0.106
	4	3,3,3,3	2,401	1,831	0	570	183.0
	5	2,2,2,2,2	243	221	8	14	4.128
	5	3,3,3,3,3	16,807	6,600	7	10,200	5,002

Table 2: Finding SWNE in NFGs (timeout of 20ms for Z3).

Implementation. CSGs are specified using the PRISM-games 3.0 modelling language, as described in [23,25]. Models are built and stored using the tool's Java-based 'explicit' engine, which employs sparse matrices. Finding SWNE of NFGs, which can be reduced to solving a nonlinear programming problem (see Section 4), is performed using a combination of the SMT solver Z3 [16] and the nonlinear optimisation suite IPOPT [39]. Although SMT solvers are able to find solutions to nonlinear problems, they are not guaranteed to do so and are only efficient in certain cases. These cases include when there is a small number of actions per player or finding support assignments for which an equilibrium is not possible. To mitigate the inefficiencies of the SMT solver, we use Z3 for filtering out unsatisfiable support assignments with a timeout: given a support assignment, Z3 returns either *unsat*, *sat* or *unknown* (if the timeout is reached). If either *sat* or *unknown* are returned, then the assignment is passed to IPOPT, which checks for satisfiability (if required) and computes SWNE values using an interior-point filter line-search algorithm [40]. To speed up the overall computation the support assignments are analysed in parallel. We also search for and filter out *dominated strategies* as a precomputation step. The NFGs are built on the fly, as well as the gradient of the objective function (3) and the Jacobian of the constraints (4)–(6), which are required as an input to IPOPT.

Table 2 presents experimental results for solving various NFGs (generated with GAMUT [31]) using Z3 (with a timeout of 20ms) and IPOPT. For each NFG, the table lists the numbers of players, actions of each player and support assignments. The table also includes the supports of each type returned by Z3 and the solution time of IPOPT. As can be seen, using Z3 significantly reduces the assignments IPOPT needs to analyse, by orders of magnitude in some cases. However, as the number of actions grows, the number of assignments that remain for IPOPT to solve increases rapidly, and therefore so does the solution time. Furthermore, increasing the number of players only magnifies this issue.

The results show that solving NFGs can be computationally very expensive. Note that just finding an NE is already a difficult problem, whereas we search for SWNE, and hence need to find *all* NE. For example, in [33], using a backtracking

Case study & property [parameters]	Players	Param. values	CSG statistics States	Max. Act.	Trans.	Constr. time(s)	Verif. time (s)
Secret Sharing $\mathtt{R}_{\max=?}[\mathtt{F}\ \mathtt{d}\vee r{=}r_{max}]$ *model*/[α,r_{max},p_{fail}]	3	*raa*/0.3,10,—	4,279	2,1,1	5,676	0.057	0.565
		rba/0.3,10,0.2	7,095	2,1,1	9,900	0.090	0.939
		rra/0.3,10,—	8,525	2,2,1	11,330	0.250	25.79
		rrr/0.3,10,—	17,017	2,2,2	22,638	0.250	96.07
Public Good $\mathtt{R}_{\max=?}[\mathtt{I}^{=k_{max}}]$ [f, k_{max}]	3	2.9,2	758	3,3,3	1,486	0.098	7.782
		2.9,3	16,337	3,3,3	36,019	0.799	110.1
		2.9,4	279,182	3,3,3	703,918	6.295	1,459
	4	2.9,1	83	3,3,3,3	163	0.046	0.370
		2.9,2	6,644	3,3,3,3	13,204	0.496	7.111
		2.9,3	399,980	3,3,3,3	931,420	11.66	99.86
	5	2.9,1	245	3,3,3,3,3	487	0.081	2.427
		2.9,2	59,294	3,3,3,3,3	118,342	2.572	2,291
Aloha (deadline) $\mathtt{P}_{\max=?}[\mathtt{F}\ \mathtt{s}_i \wedge t \leqslant D]$ [b_{max}, D]	3	1,8	3,519	2,2,2	5,839	0.168	11.23
		2,8	14,230	2,2,2	28,895	0.430	14.05
		3,8	72,566	2,2,2	181,438	1.466	18.41
		4,8	413,035	2,2,2	1,389,128	7.505	43.23
	4	1,8	23,251	2,2,2,2	42,931	0.708	75.59
		2,8	159,892	2,2,2,2	388,133	3.439	131.7
		3,8	1,472,612	2,2,2,2	4,777,924	28.69	819.2
	5	1,8	176,777	2,2,2,2,2	355,209	3.683	466.3
Aloha $\mathtt{R}_{\min=?}[\mathtt{F}\ \mathtt{s}_i]$ [b_{max}]	3	1	1,034	2,2,2	1,777	0.096	40.76
		2	5,111	2,2,2	10,100	0.210	29.36
		3	22,812	2,2,2	56,693	0.635	51.22
		4	107,799	2,2,2	355,734	2.197	150.1
Medium access $\mathtt{R}_{\max=?}[\mathtt{C}^{\leqslant k}]$ [e_{max}, k]	3	5,10	1,546	2,2,2	17,100	0.324	147.9
		10,10	10,591	2,2,2	135,915	1.688	682.7
		15,20	33,886	2,2,2	457,680	4.663	6,448
	4	5,5	15,936	2,2,2,2	333,314	4.932	3,581

Table 3: Statistics for a representative set of CSG verification instances.

search algorithm or either of the Simplicial Subdivision [38] and the Govindan-Wilson [17] algorithms for finding a sample NE, there are instances of NFGs with 6 players and 5 actions that timeout after 30 minutes.

We also comment that care needs to be taken with numerical computations. The value iteration part of the model checking algorithm is (as usual) implemented using floating point arithmetic, and may therefore exhibit small rounding errors. However, the intermediate results are passed to solvers, which may expect inputs in terms of rational numbers (Z3 in this case). It could be beneficial to investigate the use of arbitrary precision arithmetic instead.

We now present case studies and experimental results to demonstrate the applicability and performance of our approach and implementation.

Efficiency and Scalability. Table 3 presents a selection of results demonstrating the performance of the implementation. The models in the table are discussed in more detail below. The results were carried out using a 2.10GHz Intel Xeon Gold with 16GB of JVM memory. The table includes statistics for the models: number of players, states, (maximum) actions for each player in a state, transitions and the times to both build and verify the models. All models have been verified in under 2 hours and in most cases much less than this. The largest model, verified in under 15 minutes, has 4 players, almost 1.5 million states and 5 million transitions. The majority of the time is spent solving NFG games and, as shown in Table 2, this varies depending on the number of choices and players.

Secret Sharing. The first case study is the *secret sharing* protocol of [19], which uses uncertainty to induce cooperation. The protocol is defined for 3 agents and can be extended to more agents by partitioning the agents into three groups. Since the 3 agents act independently, this protocol could not be analysed with the two-coalitional variant of rPATL [23]. Each agent has an unfair coin with the same bias (α). In the first step of the protocol, agents flip their coins, and if their coins land on heads, they are supposed to send their share of the secret to the other agents. In the second step, everyone reveals the value of their coin to the other agents. The game ends if all agents obtain all shares and therefore can all reconstruct the secret, or an agent cheats, i.e., fails to send their share to another agent when they are supposed to. If neither of these conditions hold, new shares are issued to the agents and a new round starts. The protocol assumes that each agent prefers to learn the secret and that others do not learn. This is expressed by the utilities u_3, u_2, u_1 and u_0 that an agent i gets if all the agents, two agents (including i), only i and no agent is able to learn the secret, respectively.

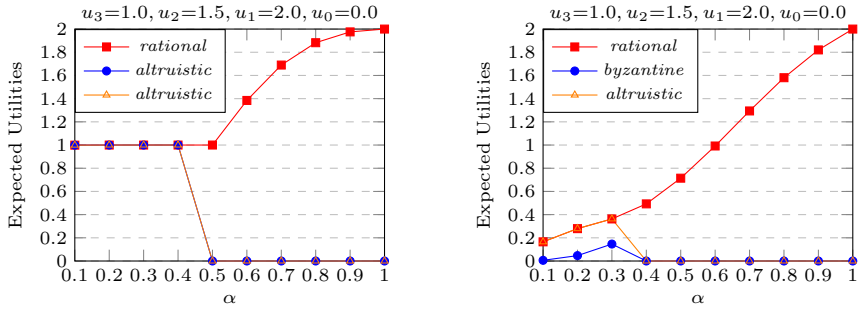
A *rational* agent in this context is one that has the choice of cheating and ignoring the coin toss in order to maximise their utility. An *altruistic* agent is one who strictly follows the protocol and a *byzantine* agent has a probability (p_{fail}) of failing and subsequently sending or computing the wrong values. Figure 1 presents the expected utilities when there are two altruistic and one rational agent and when there is one altruistic, one byzantine and one rational agent as α varies. The results when there is one altruistic and two rational agents or three rational agents yield the same graph as Figure 1(a), where the one or two additional rational agents utilities match those of the altruistic agents. According to the theoretical results of [19], for a model with one rational and two altruistic agents, the rational agent only has an incentive to cheat if:

$$(u_1 \cdot \alpha^2 + u_0 \cdot (1-\alpha)^2)/(\alpha^2 + (1-\alpha)^2) > u_3\,. \tag{7}$$

This result is validated by Figure 1(a) for the given utility values; the rational agent only cheats when $\alpha \geqslant 0.5$ (for $\alpha < 0.5$ all agents receive a utility of 1 corresponding to all agents getting the secret), which corresponds to when (7) holds for our chosen utility values. Furthermore, Figure 1 also shows that the closer α is to one then the greater the expected utility of a rational agent. Figure 1(b) also shows that, with a byzantine agent, the rational agent cheats when $\alpha \geqslant 0.4$.

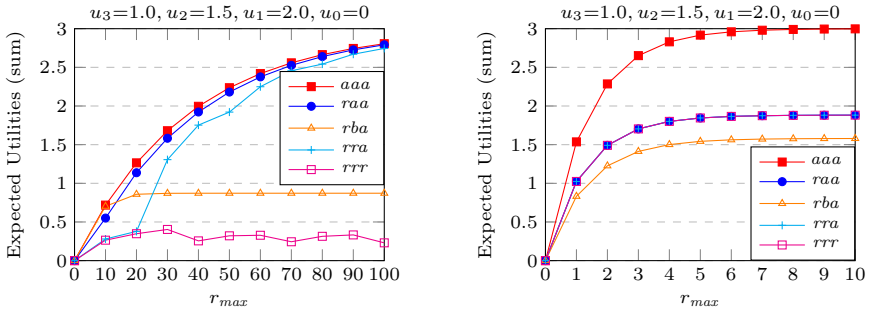
Figure 2 plots the expected utilities of the agents when the protocol stops after a maximum number of rounds (r_{max}) when $\alpha = 0.3$ and $\alpha = 0.8$. The utilities converge more slowly for $\alpha = 0.3$, since, when α is small, there is a higher chance that an agent flips tails in a round, meaning not all agents will share their secret in this round and the protocol will move into another round. Again we see that there are more incentives for a rational agent to cheat as α gets closer to 1. However, when $\alpha = 0.3$ and there are altruistic agents, the incentive decreases and eventually disappears as the number of rounds increases.

Public Good Game. We consider a variant of the public good game presented in Example 2, in which the parameter f is fixed, where each player receives an



(a) 2 altruistic and 1 rational (b) 1 altruistic, 1 byzantine and 1 rationalå

Fig. 1: $\langle\langle usr_1{:}usr_2{:}usr_3 \rangle\rangle_{\max=?}(\mathtt{R}[\mathtt{F\ done}]+\mathtt{R}[\mathtt{F\ done}]+\mathtt{R}[\mathtt{F\ done}])$ (p_{fail}=0.2).



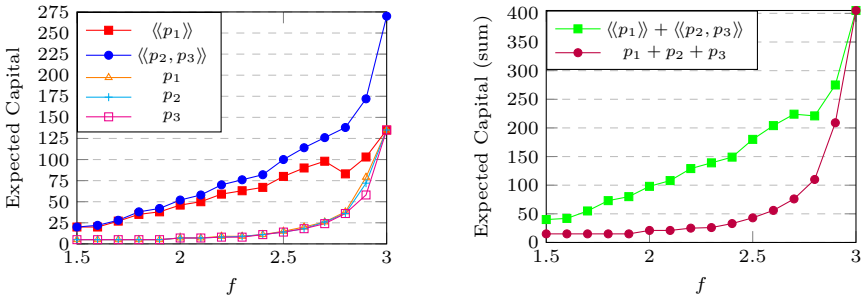
(a) α=0.3 (b) α=0.8

Fig. 2: Expected utilities over a bounded number of rounds (p_{fail}=0.2 for *rba*).

initial amount of capital (e_{init}) and, in each of k months, can invest none, half or all of their current capital. A 2-player version of the game was modelled in [25].

Figure 3 presents results for the 3-player public good game as f varies, plotting the expected utilities when the players act in isolation and, for comparison, when player 1 acts in isolation and players 2 and 3 form a coalition (indicated by $\langle\langle\rangle\rangle$), which would be required if the two-coalitional variant of rPATL [23] was used. When the players act in isolation, if $f \leqslant 2$, then there is no incentive for the players to invest. As f increases, the players start to invest some of their capital in some of the months, and when f=3 each player invests all their capital in each month. On the other hand, when players 2 and 3 act in a coalition, there is incentive to invest capital for smaller values of f, as players 2 and 3 can coordinate their investments to ensure they both profit; however, player 1 also gains from these investments, and therefore has no incentive to invest in the final month. As f increases, there is a greater incentive for player 1 to invest and the final capital for all the players increases. The drop in the capital of player 1, as f increases, is caused by players 2 and 3 coordinating against player 1 and decreasing their investments. This forces player 1 to invest to increase its investment which, as profits are shared, also increases the capital of players 2 and 3.

Aloha. This case study concerns a number of users trying to send packets using the slotted ALOHA protocol introduced in [23]. In a time slot, if a single user



(a) Individual rewards. (b) Sum of rewards.

Fig. 3: $\langle\langle p_1{:}p_2{:}p_3\rangle\rangle_{\max=?}(\mathtt{R}^{c_1}[\,\mathtt{I}^{=r_{max}}\,]+\mathtt{R}^{c_2}[\,\mathtt{I}^{=r_{max}}\,]+\mathtt{R}^{c_3}[\,\mathtt{I}^{=r_{max}}\,])$ (e_{init}=5, k=3).

tries to send a packet, there is a probability q that the packet is sent; if k users try and send, then the probability decreases to q/k. If sending a packet fails, the number of slots a user waits before resending is set according to an exponential backoff scheme. The analysis of the model in [23] consisted of considering three users with two acting in coalition. We extend the analysis by considering the case when the three act in isolation and extend the model with a fourth user. The objectives concern maximising the probability of sending a packet within a deadline, e.g. $\langle\langle usr_1{:}\cdots{:}usr_m\rangle\rangle_{\max=?}(\mathtt{P}[\,\mathtt{F}\ (\mathtt{s}_1\wedge t\leqslant D)\,]+\cdots+\mathtt{P}[\,\mathtt{F}\ (\mathtt{s}_m\wedge t\leqslant D)\,])$, and the expected time to send a packet. By allowing the users to act independently we find that the expected time required for all users to send their packets reduces compared to when two of the players act as a coalition.

Medium Access Control. This case study is based on a deterministic concurrent game model of medium access control [7]. The model consists of two users that have limited energy and share a wireless channel. The users repeatedly choose to transmit or wait and, if both transmit, the transmissions fail due to interference. We previously extended the model to three users and added the probability of transmissions failing (which is dependent on the number of users transmitting) [23]. However, the analysis was restricted to the scenario where two users were in coalition [23]. We can now remove this restriction and analyse the case when each user tries to maximise the expected number of messages they send over a bounded number of steps and extend this analysis to four users.

6 Conclusions

We have presented a logic and algorithm for model checking *multi-coalitional* equilibria-based properties of CSGs, focusing on a variant of stopping games. We have implemented the approach in PRISM-games and demonstrated its applicability on a range of case studies and properties. The main limitation of the approach is the time required for solving NFGs during value iteration as the number of players increases. Efficiency improvements that could be employed include filtering out conditionally dominated strategies [36]. Future work will also include investigating correlated equilibria [5] and mechanism design [27].

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.: Formal verification of probabilistic systems. Ph.D. thesis, Stanford University (1997)
2. de Alfaro, L., Henzinger, T., Kupferman, O.: Concurrent reachability games. Theoretical Computer Science **386**(3), 188–217 (2007)
3. de Alfaro, L., Majumdar, R.: Quantitative solution of omega-regular games. Journal of Computer and System Sciences **68**(2), 374–397 (2004)
4. Alur, R., Henzinger, T.A., Kupferman, O.: Alternating-time temporal logic. Journal of the ACM **49**(5), 672–713 (2002)
5. Aumann, R.: Subjectivity and correlation in randomized strategies. Journal of Mathematical Economics **1**(1), 67–96 (1974)
6. Bouyer, P., Markey, N., Stan, D.: Mixed Nash equilibria in concurrent games. In: Proc. FSTTCS'14, *LIPICS*, vol. 29, pp. 351–363. Leibniz-Zentrum für Informatik (2014)
7. Brenguier, R.: PRALINE: A tool for computing Nash equilibria in concurrent games. In: Proc. CAV'13, *LNCS*, vol. 8044, pp. 890–895. Springer (2013). lsv.fr/Software/praline
8. Brihaye, T., Bruyère, V., Goeminne, A., Raskin, J.F., van den Bogaard, M.: The complexity of subgame perfect equilibria in quantitative reachability games. In: Proc. CONCUR'19, *LIPICS*, vol. 140, pp. 13:1–13:16. Leibniz-Zentrum für Informatik (2019)
9. Čermák, P., Lomuscio, A., Mogavero, F., Murano, A.: MCMAS-SLK: A model checker for the verification of strategy logic specifications. In: Proc. CAV'14, *LNCS*, vol. 8559, pp. 525–532. Springer (2014)
10. 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)
11. Chatterjee, K., Henzinger, T.: Value iteration. In: 25 Years of Model Checking, *LNCS*, vol. 5000, pp. 107–138. Springer (2008)
12. 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)
13. Chen, T., Forejt, V., Kwiatkowska, M., Simaitis, A., Wiltsche, C.: On stochastic games with multiple objectives. In: Proc. MFCS'13, *LNCS*, vol. 8087, pp. 266–277. Springer (2013)
14. Cramton, P., Shoham, Y., Steinberg, R.: An overview of combinatorial auctions. SIGecom Exchanges **7**, 3–14 (2007)
15. Daskalakis, C., Goldberg, P., Papadimitriou, C.: The complexity of computing a Nash equilibrium. Communications of the ACM **52**(2), 89–97 (2009)
16. De Moura, L., Bjørner, N.: Z3: An efficient SMT solver. In: Proc. TACAS'08, *LNCS*, vol. 4963, pp. 337–340. Springer (2008). github.com/Z3Prover/z3
17. Govindan, S., Wilson, R.: A global newton method to compute Nash equilibria. Journal of Economic Theory **110**(1), 65–86 (2003)

18. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.: EVE: A tool for temporal equilibrium analysis. In: Proc. ATVA'18, *LNCS*, vol. 11138, pp. 551–557. Springer (2018). github.com/eve-mas/eve-parity
19. Halpern, J., Teague, V.: Rational secret sharing and multiparty computation: Extended abstract. In: Proc. STOC'04, pp. 623–632. ACM (2004)
20. Hauser, O., Hilbe, C., Chatterjee, K., Nowak, M.: Social dilemmas among unequals. Nature **572**, 524–527 (2019)
21. Kemeny, J., Snell, J., Knapp, A.: Denumerable Markov Chains. Springer (1976)
22. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Automated verification of concurrent stochastic games. In: Proc. QEST'18, *LNCS*, vol. 11024, pp. 223–239. Springer (2018)
23. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Equilibria-based probabilistic model checking for concurrent stochastic games. In: Proc. FM'19, *LNCS*, vol. 11800, pp. 298–315. Springer (2019)
24. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: Multi-player equilibria verification for concurrent stochastic games (2020). arXiv:2007.03365
25. Kwiatkowska, M., Norman, G., Parker, D., Santos, G.: PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In: Proc. CAV'20, LNCS. Springer (2020). To appear, prismmodelchecker.org/games
26. McKelvey, R., McLennan, A., Turocy, T.: Gambit: Software tools for game theory, version 16.0.1 (2016). gambit-project.org
27. Narahari, Y., Narayanam, R., Garg, D., Prakash, H.: Foundations of mechanism design. In: Game Theoretic Problems in Network Economics and Mechanism Design Solutions, Advanced Information and Knowledge Processing, pp. 1–131. Springer (2009)
28. von Neumann, J., Morgenstern, O., Kuhn, H., Rubinstein, A.: Theory of Games and Economic Behavior. Princeton University Press (1944)
29. Nisan, N., Roughgarden, T., Tardos, E., Vazirani, V.: Algorithmic Game Theory. CUP (2007)
30. Nocedal, J., Wächter, A., Waltz, R.: Adaptive barrier update strategies for nonlinear interior methods. SIAM Journal on Optimization **19**(4), 1674–1693 (2009)
31. 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. ACM (2004). gamut.stanford.edu
32. Osborne, M., Rubinstein, A.: An Introduction to Game Theory. OUP (2004)
33. Porter, R., Nudelman, E., Shoham, Y.: Simple search methods for finding a Nash equilibrium. In: Proc. AAAI'04, pp. 664–669. AAAI Press (2004)
34. Roughgarden, T., Tardos, E.: How bad is selfish routing? Journal of the ACM **49**, 236–259 (2002)
35. Schwalbe, U., Walker, P.: Zermelo and the early history of game theory. Games and Economic Behavior **34**(1), 123–137 (2001)
36. Shimoji, M., Watson, J.: Conditional dominance, rationalizability, and game forms. Journal of Economic Theory **83**, 161–195 (1998)
37. Toumi, A., Gutierrez, J., Wooldridge, M.: A tool for the automated verification of Nash equilibria in concurrent games. In: Proc. ICTAC'15, *LNCS*, vol. 9399, pp. 583–594. Springer (2015)
38. Van Der Laan, G., Talman, A., Van Der Heyden, L.: Simplicial variable dimension algorithms for solving the nonlinear complementarity problem on a product of unit simplices using a general labelling. Mathematics of Operations Research **12**(3), 377–397 (1987)

39. Wächter, A.: Short tutorial: Getting started with IPOPT in 90 minutes. In: Combinatorial Scientific Computing, no. 09061 in Dagstuhl Seminar Proceedings. Leibniz-Zentrum für Informatik (2009). github.com/coin-or/Ipopt
40. Wächter, A., Biegler, L.: On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. Mathematical Programming **106**(1), 25–57 (2006)
41. Supporting material. prismmodelchecker.org/files/qest20