Probabilistic Model Checking for Strategic Equilibria-based Decision Making: Advances and Challenges

Marta Kwiatkowska
University of Oxford, Oxford, UK

Gethin Norman
University of Glasgow, Glasgow, UK
University of Oxford, Oxford, UK

David Parker
University of Birmingham, Birmingham, UK

Gabriel Santos
University of Oxford, Oxford, UK

Rui Yan
University of Oxford, Oxford, UK

Abstract

Game-theoretic concepts have been extensively studied in economics to provide insight into competitive behaviour and strategic decision making. As computing systems increasingly involve concurrently acting autonomous agents, game-theoretic approaches are becoming widespread in computer science as a faithful modelling abstraction. These techniques can be used to reason about the competitive or collaborative behaviour of multiple rational agents with distinct goals or objectives. This paper provides an overview of recent advances in developing a modelling, verification and strategy synthesis framework for concurrent stochastic games implemented in the probabilistic model checker PRISM-games. This is based on a temporal logic that supports finite- and infinite-horizon temporal properties in both a zero-sum and nonzero-sum setting, the latter using Nash and correlated equilibria with respect to two optimality criteria, social welfare and social fairness. We summarise the key concepts, logics and algorithms and the currently available tool support. Future challenges and recent progress in adapting the framework and algorithmic solutions to continuous environments and neural networks are also outlined.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Probabilistic model checking, stochastic games, equilibria

Digital Object Identifier 10.4230/LIPIcs.MFCS.2022.4

Category Invited Talk

Funding This project was funded by the ERC under the European Union's Horizon 2020 research and innovation programme (FUN2MODEL, grant agreement No. 834115).

1 Introduction

Game-theoretic techniques have long been a source of fundamental insights into strategic decision making for multi-agent systems. They have been widely studied in areas such as economics [27], control [43] and robotics [36]. Concurrent stochastic multi-player games (CSGs), in particular, provide a natural framework for modelling a set of interactive, rational agents operating concurrently within an uncertain or stochastic environment. They can be viewed as a collection of players (agents) with strategies for determining their actions based on the execution so far, and where the resulting evolution of the system is probabilistic.

47th International Symposium on Mathematical Foundations of Computer Science (MFCS 2022).
Editors: Stefan Szeider, Robert Ganian, and Alexandra Silva; Article No. 4; pp. 4:1–4:22
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Game-theoretic analysis is versatile, in that it can support both zero-sum and nonzero-sum (equilibria) analysis. Zero-sum properties focus on scenarios in which one player (or a coalition of players) aims to optimise some objective, while the remaining players form a coalition with the directly opposing goal. On the other hand, nonzero-sum (equilibria) properties correspond to situations where two or more players (or coalitions of players) in a CSG have distinct objectives to be maximised or minimised. In nonzero-sum properties the goals of the players (or coalitions) are not necessarily directly opposing, and therefore it may be beneficial for players to collaborate. Competitive scenarios occur in many applications, e.g., attackers and defenders in the context of computer security. Similarly, collaborative behaviour can be essential, e.g., to effectively control a multi-robot system, or for users to send data efficiently through a shared medium in a communication protocol.

Probabilistic model checking is a powerful approach to the formal analysis of systems with stochastic behaviour. It relies on the construction and analysis of a probabilistic model, guided by a formal specification of its desired behaviour in temporal logic. It is of particular benefit in the context of models, such as stochastic games, which combine nondeterministic and probabilistic behaviour. This is because the interplay between these aspects of the model can be subtle and lead to unexpected results if not carefully modelled and analysed. This is exacerbated when the system comprises multiple agents with differing objectives.

Until recently, practical applications of probabilistic model checking based on stochastic games had focused primarily on *turn-based* models [15], in which simultaneous decision making by agents is forbidden. Alternatively, model checking of *non-stochastic* games has been extensively studied, and tool support developed [5, 39]. CSGs provide a more powerful and realistic modelling formalism, but also bring considerable challenges, in terms of the higher computational complexity or undecidability for some key problems.

There has nonetheless been significant amounts of work on tackling verification problems for CSGs. A number of algorithms have been proposed for solving CSGs against formally specified zero-sum properties, e.g. [17, 18, 11]. In the case of nonzero-sum properties, [14, 25] study the existence of and the complexity of finding equilibria for stochastic games. Complexity results for finding equilibria are also considered in [9] and [23] for quantitative reachability properties and temporal logic properties, respectively. Other work concerns finding equilibria for discounted properties; we mention [49], which formulates a learning-based algorithm, and [40], which presents iterative algorithms. However these advances are mostly lacking in implementations, tool support or case studies. Tools exist for solving turn-based stochastic games [13, 16] and non-stochastic concurrent games [16, 8, 10, 55, 24, 45], with the latter class including support for computing equilibria.

At the same time, there is an increasing trend to incorporate data-driven decision making, which necessitates the incorporation on of machine learning components within autonomous systems, which are built largely using conventional, symbolic methods. Examples of such *neuro-symbolic* systems are self-driving cars whose vision function is provided via a neural network image classifier, or an aircraft controller whose collision avoidance system uses a neural network for decision support. Design automation support for such systems is lacking, yet automatic computation of equilibria aids in ensuring stable solutions.

This paper provides an overview of recent advances in developing a modelling, verification and strategy synthesis framework for concurrent stochastic games, as implemented in the PRISM-games probabilistic model checker [33]. The framework uses a temporal logic that supports a wide range of finite- and infinite-horizon properties, relating to the probability of an event's occurrence or the expected amount of reward or cost accumulated. The logic allows specification of both *zero-sum* and *nonzero-sum* properties, with the latter expressed using either *Nash equilibria* or *correlated equilibria*. For both types of equilibria, strategies are synthesised in which it is not beneficial for any player to unilaterally alter their chosen actions, but correlated equilibria also allow players to coordinate through *public signals*. Since several, varied such equilibria may exist, we also support distinct optimality criteria to select between them; we consider *social welfare*, which maximises the sum of the players utilities, and *social fairness*, which minimises the difference between the utilities.

We summarise the key concepts, logics and algorithms that underlie this framework and discuss the tool support provided by PRISM-games, including an illustrative case study of formally modelling and analysing a multi-agent communication protocols using CSGs. Future challenges and recent progress in extending the framework and algorithmic solutions to modelling of neuro-symbolic CSGs are also outlined. In contrast to the majority of prior research, the focus of this strand of work is on software tool development, applications and case studies.

2 Normal form games

We introduce the main concepts used in this paper by means of simple one-shot games known as *normal form games* (NFGs), where players make their choices at the same time. We consider both zero-sum NFGs and nonzero-sum NFGs, then define equilibria concepts for these games and summarise existing algorithms for equilibria computation.

We first require the following notation. Let $\mathit{Dist}(X)$ denote the set of probability distributions over set X. For any vector $v \in \mathbb{R}^n$, we use $v(i)$ to refer to the ith entry of the vector. For any tuple $x = (x_1, \ldots, x_n) \in X^n$, element $x' \in X$ and $i \leqslant n$, we define the tuples $x_{-i} \overset{\text{def}}{=} (x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ and $x_{-i}[x'] \overset{\text{def}}{=} (x_1, \ldots, x_{i-1}, x', x_{i+1}, \ldots, x_n)$.

▶ **Definition 1** (Normal form game). *A (finite, n-player)* 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* $u_i \colon A \to \mathbb{R}$ *is a* utility function *for player* $i \in N$.

In a normal form game N, the players choose actions simultaneously, with player $i \in N$ choosing an action from the set A_i and, assuming that each player $i \in N$ selects action a_i, player j receives the utility $u_j(a_1, \ldots, a_n)$. The *objective* of each player is to maximise their utility and their choices are governed by *strategies*, which we now define. We will also distinguish *strategy profiles*, which comprise a strategy for each player, and *correlated profiles*, which correspond to choices of the players when they are allowed to coordinate through a (probabilistic) *public signal*.

▶ **Definition 2** (Strategies, profiles and correlated profiles). *For an NFG* N*:*

- *a* strategy σ_i *for player* i *in an NFG* N *is a probability distribution over the set of actions* A_i *and we let* Σ^i_{N} *denote the set of all strategies for player* i*;*
- *a* strategy profile *(or* profile*)* $\sigma = (\sigma_1, \ldots, \sigma_n)$ *is a tuple of strategies for each player;*
- *a* correlated profile *is a tuple* (τ, ς) *comprising* $\tau \in \mathit{Dist}(D_1 \times \cdots \times D_n)$*, where* D_i *is a finite set of* signals *for player* i*, and* $\varsigma = (\varsigma_1, \ldots, \varsigma_n)$*, where* $\varsigma_i \colon D_i \to A_i$ *is a function from the signals of player* i *to the actions of player* i*.*

For a correlated profile (τ, ς) of N, the public signal τ is a joint distribution over signals D_i for each player i such that, if player i receives the signal $d_i \in D_i$, then it chooses action

$\varsigma_i(d_i)$. We can consider any correlated profile (τ, ς) as a *joint strategy*, i.e., a distribution over $A_1 \times \cdots \times A_n$ where:

$$(\tau, \varsigma)(a_1, \ldots, a_n) = \sum\{\tau(d_1, \ldots, d_n) \mid d_i \in D_i \wedge \varsigma(d_i) = a_i \text{ for all } i \in N\}\,.$$

Conversely, any joint strategy $\tau \in \mathit{Dist}(A_1 \times \cdots \times A_n)$ of N can be considered as a correlated profile (τ, ς), where $D_i = A_i$ and ς_i is the identity function for $i \in N$. Any profile σ of an NFG N can be mapped to an equivalent correlated profile (in which τ is the joint distribution $\sigma_1 \times \cdots \times \sigma_n$ and ς_i is the identity function). On the other hand, there are correlated profiles with no equivalent strategy profile.

Under profile σ or correlated profile (τ, ς) the expected utilities of player i are:

$$\begin{array}{rcl} u_i(\sigma) & \stackrel{\text{def}}{=} & \sum_{(a_1,\ldots,a_n) \in A} u_i(a_1, \ldots, a_n) \cdot \left(\prod_{j=1}^{n} \sigma_j(a_j)\right) \\ u_i(\tau, \varsigma) & \stackrel{\text{def}}{=} & \sum_{(d_1,\ldots,d_n) \in D} \tau(d_1, \ldots, d_n) \cdot u_i(\varsigma_1(d_1), \ldots, \varsigma_n(d_n))\,. \end{array}$$

▶ **Example 3.** Consider the two-player NFG with available action sets $A_i = \{\mathit{heads}_i, \mathit{tails}_i\}$ for $1 \leqslant i \leqslant 2$ and a correlated profile corresponding to the joint distribution $\tau \in \mathit{Dist}(A_1 \times A_2)$, where $\tau(\mathit{heads}_1, \mathit{heads}_2) = \tau(\mathit{tails}_1, \mathit{tails}_2) = 0.5$. Under this correlated profile, the players share a fair coin and choose their action based on the outcome of the coin toss. There is no equivalent strategy profile.

2.1 Zero-sum NFGs

A *zero-sum NFG* is a two-player NFG N such that $u_1(\alpha) + u_2(\alpha) = 0$ for all $\alpha \in A$, meaning that the objectives of the players are directly opposing. Such an 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 $A_1 = \{a_1, \ldots, a_l\}$, $A_2 = \{b_1, \ldots, b_m\}$ and $z_{ij} = u_1(a_i, b_j) = -u_2(a_i, b_j)$.

We next introduce the notion of the *value* of a zero-sum NFG and recall classical results about the existence of optimal strategies.

▶ **Theorem 4** (Minimax theorem [56, 57]). *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., $\inf_{\sigma_2 \in \Sigma_{\mathsf{N}}^2} u_1(\sigma_1^\star, \sigma_2) \geqslant v^\star$;*
- *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., $\inf_{\sigma_1 \in \Sigma_{\mathsf{N}}^1} u_2(\sigma_1, \sigma_2^\star) \geqslant -v^\star$.*

The value of a matrix game $\mathsf{Z} \in \mathbb{Q}^{l \times m}$ can be found by solving a linear programming (LP) problem [56, 57].

▶ **Example 5.** Table 1 shows a classic example of a two-player zero-sum game known as *matching pennies*. Columns α and u_i represent the collective choice (profile) and player i's utility, respectively. In this example, each player has a coin for which they may choose the value to be heads or tails, i.e., $A_i = \{\mathit{heads}_i, \mathit{tails}_i\}$. If the coins match, player 1 wins the round, which is indicated by being awarded a utility of 1, while player 2 receives utility -1. If the coins do not match, then the players' utilities are negated.

α	$u_1(\alpha)$	$u_2(\alpha)$	α	$u_1(\alpha)$	$u_2(\alpha)$
$(heads_1, heads_2)$	1	-1	$(tails_1, heads_2)$	-1	1
$(heads_1, tails_2)$	-1	1	$(tails_1, tails_2)$	1	-1

Table 1 Matching pennies game in normal form.

The value for the corresponding matrix game is the solution to the following LP problem: Maximise v subject to:

$$x_1 - x_2 \geqslant v,\ x_2 - x_1 \geqslant v,\ x_1 + x_2 = 1$$

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

2.2 Nonzero-sum NFGs

The requirement for players to have directly opposing objectives is often too limiting, and it is necessary to allow distinct objectives, which cannot be modelled in a zero-sum fashion. These scenarios can be captured using the notion of *equilibria*, defined by a separate, independent objective for each agent. We now define the concepts of *Nash equilibrium* [57] and *correlated equilibrium* [6] for NFGs, which ensure stability against deviations by individual agents, improving the overall game outcomes. Since many equilibria may exist, we also introduce optimality criteria for these equilibria: *social welfare*, which is standard [46], and *social fairness*, which was first defined in [35].

Before giving the formal definitions, we first extend our notation as follows: for any profile σ and strategy $\sigma_i^\star$, the strategy tuple σ_{-i} corresponds to σ with the strategy of player i removed and $\sigma_{-i}[\sigma_i^\star]$ to the profile σ after replacing player i's strategy with $\sigma_i^\star$.

▶ **Definition 6** (Best response). *For any nonzero-sum NFG N and profile σ or correlated profile (τ, ς) of N, the* best response *moves for player i to σ_{-i} and (τ, ς_{-i}) are, respectively:*

- *a strategy $\sigma_i^\star$ for player i such that $u_i(\sigma_{-i}[\sigma_i^\star]) \geqslant u_i(\sigma_{-i}[\sigma_i])$ for all $\sigma_i \in \Sigma^i_{\mathsf{N}}$;*
- *a function $\varsigma_i^\star \colon D_i \to A_i$ for player i such that $u_i(\tau, \varsigma_{-i}[\varsigma_i^\star]) \geqslant u_i(\tau, \varsigma_{-i}[\varsigma_i])$ for all functions $\varsigma_i \colon D_i \to A_i$.*

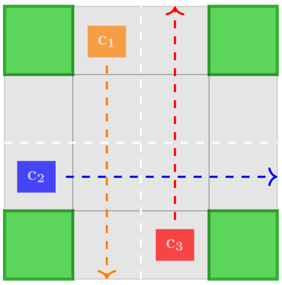
▶ **Definition 7** (NE and CE). *For any nonzero-sum NFG N, a strategy profile $\sigma^\star$ is a* Nash equilibrium *(NE) and a correlated profile $(\tau, \varsigma^\star)$ of N is a* correlated equilibrium *(CE) if:*

- *$\sigma_i^\star$ is a best response to $\sigma_{-i}^\star$ for all $i \in N$;*
- *$\varsigma_i^\star$ is a best response to $(\tau, \varsigma_{-i}^\star)$ for all $i \in N$;*

respectively.

Any NE of N is also a CE, while there exist CEs that cannot be represented by a strategy profile, and therefore are not NEs. For each class of equilibria, NE and CE, we introduce two optimality criteria, the first maximising *social welfare* (SW), defined as the *sum* of the utilities, and the second maximising *social fairness* (SF), which minimises the *difference* between the players' utilities. Other variants of fairness have been considered for NEs, such as in [38], where the authors seek to maximise the lowest utility among the players.

▶ **Definition 8** (SW and SF). *An equilibrium $\sigma^\star$ is a* social welfare *(SW) equilibrium if the sum of the utilities of the players under $\sigma^\star$ is maximal over all equilibria, while $\sigma^\star$ is a* social fair *(SF) equilibrium if the difference between the player's utilities under $\sigma^\star$ is minimised over all equilibria.*



α	$u_1(\alpha)$	$u_2(\alpha)$	$u_3(\alpha)$
(pro_1, pro_2, pro_3)	-1000	-1000	-100
(pro_1, pro_2, yld_3)	-1000	-100	-5
(pro_1, yld_2, pro_3)	5	-5	5
(pro_1, yld_2, yld_3)	5	-5	-5
(yld_1, pro_2, pro_3)	-5	-1000	-100
(yld_1, pro_2, yld_3)	-5	5	-5
(yld_1, yld_2, pro_3)	-5	-5	5
(yld_1, yld_2, yld_3)	-10	-10	-10

Figure 1 Example from [35]: Cars at an intersection and the corresponding NFG.

We can also define the dual concept of *social cost* (SC) equilibria [34], where players try to minimise, rather than maximise, their expected utilities by considering equilibria of the game $\mathsf{N}^- = (N, A, -u)$ in which the utilities of N are negated. We remark that SC equilibria strategies are not a subset of classically defined NE or CE strategies of N.

▶ **Example 9.** Consider the scenario from [35], based on an example from [50], where three cars meet at an intersection and want to proceed as indicated by the arrows in Figure 1. Each car can either *proceed* or *yield*. If two cars with intersecting paths proceed, then there is an accident. If an accident occurs, the car having the right of way, i.e., the other car is to its left, has a utility of -100 and the car that should yield has a utility of -1000. If a car proceeds without causing an accident, then its utility is 5 and the cars that yield have a utility of -5. If all cars yield, then, since this delays all cars, all have utility -10. The 3-player NFG is given in Figure 1. The different optimal equilibria of the NFG are:

- the SWNE and SWCE are the same: for c_2 to yield and c_1 and c_3 to proceed, with the expected utilities of the players $(5, -5, 5)$;
- the SFNE is for c_1 to yield with probability 1, c_2 to yield with probability 0.863636 and c_3 to yield with probability 0.985148, with the expected utilities of the players $(-9.254050, -9.925742, -9.318182)$;
- the SFCE gives a joint distribution where the probability of c_2 yielding and of c_1 and c_3 yielding are both 0.5 with the expected utilities of the players $(0, 0, 0)$.

Modifying u_2 such that $u_2(pro_1, pro_2, pro_3) = -4.5$ to, e.g., represent a reckless driver, the SWNE becomes for c_1 and c_3 to yield and c_2 to proceed with the expected utilities of the players $(-5, 5, -5)$, while the SWCE is still for c_2 to yield and c_1 and c_3 to proceed. The SFNE and SFCE also do not change.

Algorithms for computing equilibria in NFGs. Finding NEs in two-player NFGs is in the class of *linear complementarity* problems (LCPs). Established algorithms include the Lemke-Howson algorithm [37], which is based on the method of labelled polytopes [46], support enumeration [48] and regret minimisation [52]. In [34] a method for NE computation is developed, which reduces the problem to SMT via labelled polytopes [46] by considering the regions of the strategy profile space. This method iteratively reduces the search space of profiles as positive probability assignments are found and added as constraints on the profiles. This approach can also be used for finding both an SWNE and SFNE by computing all NEs and then selecting an optimal one.

In the case of NFGs with more than two players, the computation of NEs is more complex since, for a given support (i.e., a sub-region of the strategy profile space which fixes the set of actions chosen with nonzero probability by each player), finding NEs cannot be reduced to an LP problem. A method for such NFGs is presented in [32], based on support enumeration [48], which exhaustively examines all supports one at a time, checking whether that sub-region contains NEs. For each support, finding an SWNE can be reduced to a *nonlinear programming problem* [32]. This nonlinear programming problem can be modified to find an SFNE in each support [35].

In the case of CEs, the approach introduced in [35] is to first find a joint strategy for the players, i.e., a distribution over the action tuples, which can then be mapped to a correlated profile. For SWCEs, [35] reduces the computation to solving a LP problem which has $|A|$ variables, one for each action tuple, and $\sum_{i \in N}(|A_i|^2 - |A_i|) + |A| + 1$ constraints. For SFCEs, on the other hand, the method of [35] involves solving an optimisation problem with an additional has $|N| + 2$ variables and $3 \cdot |N|$ constraints compared to the LP problem for finding SWCEs.

3 Concurrent Stochastic Games

This section introduces *concurrent stochastic games* (CSGs) [54], in which players repeatedly make simultaneous choices over actions and the action choices cause a probabilistic update of the game state. CSGs thus provide a natural framework for modelling a set of interactive, rational agents operating concurrently within an uncertain or probabilistic environment. Compared to normal form games, they are classified as *multi-stage*, which is more convenient for specifying repeated or sequential interactions among agents. The introduction of stochasticity facilitates modelling of a wide range of important phenomena, for example uncertain behaviour due to noisy sensors or unreliable hardware in a multi-robot system, or the use of randomisation for coordination in a distributed security or networking protocol.

▶ **Definition 10** (Concurrent stochastic game). *A* concurrent stochastic multi-player game *(CSG) is a tuple* $\mathsf{G} = (N, S, \bar{s}, A, \Delta, \delta)$ *where:*

- $N = \{1, \dots, n\}$ *is a finite set of players;*
- S *is a finite set of states and* $\bar{s} \in S$ *is an initial state;*
- $A = (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$ *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.*

Given a CSG G, the set of actions available to player $i \in N$ in state $s \in S$ is given by $A_i(s) \overset{\text{def}}{=} \Delta(s) \cap A_i$. The CSG G starts in the initial state $\bar{s}$ and, if G is in the game state s, then each player $i \in N$ selects an action from its available actions in state s if this set is non-empty, and from $\{\bot\}$ otherwise. Next, supposing each player $i \in N$ chooses action a_i, the game state is updated according to the distribution $\delta(s, (a_1, \dots, a_n))$. We allow sets of players $C \subseteq N$ to form *coalitions*, and will consider the induced CSG, called the *coalition game*, with coalitions as players.

A *path* π of a CSG 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 $\mathit{FPaths}_{\mathsf{G},s}$ and $\mathit{IPaths}_{\mathsf{G},s}$ the sets of finite and infinite paths starting in state s of G, respectively, and drop the subscript s when considering all finite and infinite paths of G. As for NFGs, we can define *strategies* of G that resolve the choices of the players. Here, a strategy for player i is a function $\sigma_i \colon \mathit{FPaths}_{\mathsf{G}} \to \mathit{Dist}(A_i \cup \{\bot\})$ mapping finite paths to distributions over available actions, such that, if $\sigma_i(\pi)(a_i) > 0$, then $a_i \in A_i(\mathit{last}(\pi))$ where $\mathit{last}(\pi)$ is the final state of π. Furthermore, we can define strategy profiles, correlated profiles and joint strategies analogously to Section 2.

A *labelled* CSG is a tuple (G, AP, L), where G is a CSG, as in Definition 10, AP is a set of atomic propositions and $L\colon S \to 2^{AP}$ is a labelling function, specifying which atomic propositions are true in each state. We also associate CSGs with *reward structures*, which annotate states and transitions with real values. More precisely a reward structure is a pair $r{=}(r_A, r_S)$ consisting of an action reward function $r_A\colon S{\times}A \to \mathbb{R}$ and state reward function $r_S\colon S \to \mathbb{R}$. We use atomic propositions and rewards as the building blocks to specify players' utilities in a CSG, which will be described in Section 4.

Formally, the utility function or *objective* of player i in a CSG is given by a random variable $X_i\colon \mathit{IPaths}_{\mathsf{G}} \to \mathbb{R}$ over infinite paths. For a profile σ and state s, using standard techniques [31], we can construct a probability measure $\mathit{Prob}^{\sigma}_{\mathsf{G},s}$ over the paths that start in state s corresponding to σ, denoted $\mathit{IPaths}^{\sigma}_{\mathsf{G},s}$, and define the expected value $\mathbb{E}^{\sigma}_{\mathsf{G},s}(X_i)$ of player i's utility from s under σ. Similarly, we can also define such a probability measure and expected value given a correlated profile or joint strategy of G.

3.1 Zero-sum CSGs

Similarly to NFGs (see Section 2.1), *zero-sum* CSGs are two-player games that have a single utility function X for player 1, with the utility function of player 2 given by $-X$, and both players aiming to maximise the expected value of their utility. Equivalently, we can suppose that player 1 tries to maximise the expected value of X, while player 2 tries to minimise it. As for NFGs (see Theorem 4), a CSG has a *value* 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 when starting from any state of the CSG. Since the CSGs we discuss in this paper are finite-state and finitely-branching, it follows that they are determined for all of the objectives that we consider [44].

Given a multi-player CSG and objective X, we can divide the players into two coalitions, $C \subseteq N$ and $N \backslash C$, and then construct a *two-player* zero-sum coalition game, in which each coalition acts as a single player, with one coalition trying to maximise the value of X and the other trying to minimise that value.

3.2 Nonzero-sum CSGs

We define *nonzero-sum* CSGs similarly to NFGs: we assume that there is a distinct and independent objective X_i for each player i (or coalitions of players). We can then define NE and CE for CSGs (see Definition 7), as well as the restricted classes of SW and SF equilibria, similarly to those for NFGs (see Definition 8). Following [34, 32], we focus on *subgame-perfect* equilibria [47], which are equilibria in *every state* of G. Furthermore, because we include infinite-horizon objectives, where the existence of NE is an open problem [7], we will in some cases use ε-NE, which do exist for any $\varepsilon > 0$ for all the infinite-horizon objectives 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 $\mathbb{E}^{\sigma^\star}_{G,s}(X_i) \geqslant \sup_{\sigma_i \in \Sigma_i} \mathbb{E}^{\sigma^\star_{-i}[\sigma_i]}_{G,s}(X_i) - \varepsilon$ for all $i \in N$ and $s \in S$.*

▶ **Example 12.** As an example scenario that can be modelled as a CSG, consider a number of users trying to send packets using the slotted ALOHA protocol studied in [32, 34, 35]. If there is a collision or if sending a packet fails, a user waits for some number of slots before resending, with the wait set according to an exponential backoff scheme.

If we model this scenario as a CSG then, when a player has a packet to send, the actions available to the player correspond to either sending their packet or waiting to send the packet at some future time step. In the case when one coalition of players has an objective related to sending their packets efficiently, e.g., minimising the expected time to send their packets, and the remaining players form a second coalition and have the dual objective, we can model this scenario as a zero-sum CSG. In such a zero-sum CSG, the optimal strategy for the first coalition is to try and choose times to send that avoid collisions, while the second coalition will do the opposite and instead try and cause collisions. On the other hand, when there are more coalitions and each coalition's goal corresponds to sending their own packets efficiently, we can model this as a nonzero-sum CSG. Here we would be looking for equilibria, i.e., profiles such that no coalition could improve its objective by changing its strategy, which are also optimal, e.g., the sum of the expected times is minimal or the difference between the expected time to send for each coalition is minimal.

4 Property specifications and model checking for CSGs

Probabilistic model checking is a technique for systematically constructing a stochastic model and analysing it against a quantitative property formally specified in temporal logic. This approach can be used either to *verify* that a specification is always satisfied or to perform *strategy synthesis*, i.e., to construct a witness to the satisfaction of a property. In the context of CSGs, the latter means synthesising strategies for one or more players (or coalitions) such that the resulting behaviour of the game satisfies the specification.

To specify properties of labelled CSGs, we use the property specification language of the PRISM-games model checker [33], which is based on the logic PCTL (probabilistic computation tree logic) [26], extended with operators to specify expected reward properties [21] and the *coalition* operator $\langle\langle C \rangle\rangle$ from alternating temporal logic (ATL) [4]. The variant of this logic that just considers *zero-sum* formulae is referred to as rPATL (probabilistic alternating-time temporal logic with rewards) in [15], but here we use a further extended version that also supports *nonzero-sum* properties, using the notion of equilibria [34, 35].

▶ **Definition 13** (PRISM-games logic [34, 35]). *The syntax of the PRISM-games logic is given by the grammar:*

$$\begin{array}{lcl}
\phi & \coloneqq & \mathtt{true} \mid \mathsf{a} \mid \neg\phi \mid \phi\wedge\phi \mid \langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,] \mid \langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\rho\,] \mid \langle\langle \mathbb{C}\rangle\rangle(\star_1,\star_2)_{\mathrm{opt}\sim x}(\theta) \\
\psi & \coloneqq & \mathtt{X}\,\phi \mid \phi\;\mathtt{U}^{\leqslant k}\;\phi \mid \phi\;\mathtt{U}\;\phi \\
\rho & \coloneqq & \mathtt{I}^{=k} \mid \mathtt{C}^{\leqslant k} \mid \mathtt{F}\;\phi \\
\theta & \coloneqq & \mathtt{P}[\,\psi\,]+\cdots+\mathtt{P}[\,\psi\,] \mid \mathtt{R}^{r}[\,\rho\,]+\cdots+\mathtt{R}^{r}[\,\rho\,]
\end{array}$$

where a *is an atomic proposition,* $\mathbb{C} = C_1{:}\cdots{:}C_m$*,* C *and* $C_1,\ldots,C_m$ *are coalitions of players such that* $C' = N\backslash C$*,* $C_i \cap C_j = \varnothing$ *for all* $1 \leqslant i \neq j \leqslant m$*,* $(\star_1,\star_2) \in \{\textsc{ne},\textsc{ce}\}\times\{\textsc{sw},\textsc{sf}\}$*,* $\mathrm{opt} \in \{\min,\max\}$*,* $\sim\,\in \{<,\leqslant,\geqslant,>\}$*,* $q \in \mathbb{Q}\cap[0,1]$*,* $x \in \mathbb{Q}$*,* r *is a reward structure and* $k \in \mathbb{N}$*.*

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. Sums of formulae (θ) are used to specify multiple objectives for equilibria.

We omit the formal semantics, which can be found in [34, 35]. Path and reward formulae are used to express the utilities of the players, i.e., random variables over paths. For path formulae, we 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*). The random variable corresponding to the path formula ψ returns 1 for paths that satisfy ψ and zero otherwise. For reward formulae, 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 random variable corresponding to the reward formula ρ returns for a path the reward corresponding to ρ.

4.1 Zero-sum formulae

A state satisfies a formula $\langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,]$ if the coalition of players $C \subseteq N$ 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.

The model checking algorithms presented in [34] involve graph-based analysis followed by backward induction [53, 57] for exact computation of *finite-horizon properties* and value iteration [51, 12] for approximate computation of *infinite-horizon properties.* During both backward induction and value iteration for each state, at each iteration, an LP problem of size $|A|$ must be solved (corresponding to finding the value of a zero-sum one-shot game), which has complexity PTIME [30].

Strategy synthesis for the formulae $\langle\langle C\rangle\rangle\mathtt{P}_{\sim q}[\,\psi\,]$ and $\langle\langle C\rangle\rangle\mathtt{R}^{r}_{\sim x}[\,\rho\,]$ corresponds to finding optimal strategies for the players in coalition C when their objective, respectively, is maximising the probability of satisfying the formula ψ and maximising the expected value of the reward formula with respect to the reward structure r. All strategies synthesised are randomised and can be found during model checking by extracting not just the value of the zero-sum one-shot game solved in each state, but also an optimal (randomised) strategy. For infinite-horizon objectives, the synthesised strategies are memoryless, while for finite-horizon objectives, the synthesised strategies are finite-memory, with a separate distribution required for each state and each time step.

4.2 Nonzero-sum formulae

Nonzero-sum formulae allow us to reason about equilibria, for either of the types (NE or CE) and optimality criteria (SW or SF) considered here. A probabilistic formula $\langle\langle C_1{:}\cdots{:}C_m\rangle\rangle(\star_1,\star_2)_{\max\sim x}(\mathtt{P}[\,\psi_1\,]{+}\cdots{+}\mathtt{P}[\,\psi_m\,])$ is true in a state if, when the players form the coalitions $C_1,\ldots,C_m$, there is a subgame-perfect equilibrium of type $\star_1$ meeting the optimality criterion $\star_2$ 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 objective of coalition C_i is to maximise the probability of satisfying a path formula ψ_i.

For a reward formula $\langle\langle C_1{:}\cdots{:}C_m\rangle\rangle(\star_1,\star_2)_{\max\sim x}(\mathtt{R}^{r_1}[\,\rho_1\,]{+}\cdots{+}\mathtt{R}^{r_m}[\,\rho_m\,])$ the meaning is similar; however, here the objective of coalition C_i refers to a reward formula ρ_i with respect to reward structure r_i. Formulae of the form $\langle\langle C_1{:}\cdots{:}C_m\rangle\rangle(\star_1,\star_2)_{\min\sim x}(\theta)$ correspond to the dual notion of cost equilibria, which are also supported. We also allow *numerical* queries of the form $\langle\langle C_1{:}\cdots{:}C_m\rangle\rangle(\star_1,\star_2)_{\mathrm{opt}=?}(\theta)$, which return the sum of the subgame-perfect equilibrium's values of of type $\star_1$ meeting the optimality criterion $\star_2$.

Model checking algorithms, presented in [34, 32, 35], involve solving an m-player *coalition game* $\mathsf{G}^{\mathcal{C}}$, where $\mathcal{C} = \{C_1,\ldots,C_m\}$ and the choices of each player i in $\mathsf{G}^{\mathcal{C}}$ correspond to the choices of the players in coalition C_i in G. If all the objectives in θ are finite-horizon, then *backward induction* [53, 57] can be applied to compute (precise) optimal equilibria values. On the other hand, if all the objectives are infinite-horizon, *value iteration* [12] can be used to approximate optimal equilibria values. When there is a combination of finite- and infinite-horizon objectives, the game under study is modified in a standard manner to make all objectives infinite-horizon.

Both backward induction and value iteration over the CSG $\mathsf{G}^{\mathcal{C}}$ work by iteratively computing new values for each state s of $\mathsf{G}^{\mathcal{C}}$. The values for each state, in each iteration, are found by computing optimal equilibria values, with respect to the criterion $\star_2$ and equilibrium type $\star_1$, of an NFG N whose utility function is derived from the outgoing transition probabilities from s in the CSG and the values computed for successor states of s in the previous iteration.

We can synthesise a strategy profile representing the appropriate type of equilibrium for the CSG by combining the optimal strategies for the equilibria generated in each individual state during solution. As for zero-sum formulae, randomisation is required and memory is needed both to keep track of both the step bound of finite-horizon objectives and the satisfaction of each player's objective.

▶ **Example 14.** We now return to the scenario from Example 12, where a number users are attempting to send packets using the slotted ALOHA protocol. The zero-sum formulae $\langle\!\langle usr_1, \dots, usr_k \rangle\!\rangle \mathtt{R}^{time}_{\min=?}[\,\mathtt{F}\ \mathsf{sent}_{1\dots k}\,]$ and $\langle\!\langle usr_1, \dots, usr_k \rangle\!\rangle \mathtt{P}_{\max=?}[\,\mathtt{F}\ \mathsf{sent}_{1\dots k} \wedge t \leqslant D\,]$ represent the case where the first k users form a coalition and try to minimise the expected time to send their packets or maximise the probability they send their packets within a deadline $D \in \mathbb{N}$, respectively, while the remaining users form a second coalition and try and achieve the opposite objective, i.e., maximise the expected time or minimise the probability.

On the other hand, in the nonzero-sum case, if we suppose there are m users and the objective of each user is to minimise the expected time to send their packet, this can be expressed by the nonzero-sum formula $\langle\!\langle usr_1 {:} \cdots {:} usr_m \rangle\!\rangle (\star_1, \star_2)_{\min=?}(\mathtt{R}^{time}[\,\mathtt{F}\ \mathsf{sent}_1\,] + \cdots + \mathtt{R}^{time}[\,\mathtt{F}\ \mathsf{sent}_m\,])$.

5 Tool support and case studies

Tool support for the modelling and automated verification of CSGs has been implemented in PRISM-games [33], which is available from [61]. A variety of case studies have been modelled and analysed as CSGs with the tool, using both zero-sum and nonzero-sum properties. These include: a robot coordination problem [34]; futures market investors [34, 35]; medium access control [32, 34]; power control [34, 35]; a public good game [32, 35] and secret sharing [32]. The results for these case studies demonstrate: the advantages of using CSGs for modelling (for example, with respect to simpler turn-based games); that using nonzero-sum properties can yield gains for the players (or coalitions); and that the use of correlated equilibria and social fairness results may be advantageous compared to Nash equilibria and social welfare. We give a brief description of the functionality and implementation of PRISM-games and then present a representative case study: the slotted ALOHA protocol.

5.1 PRISM-games

PRISM-games [33] is an extension of the PRISM model checker, which provides support for a variety of stochastic game models, including turn-based and concurrent multi-player stochastic games, and (turn-based) timed probabilistic games.

These are all described in the PRISM-games modelling language, a stochastic extension of the Reactive Modules formalism [3]. The language facilitates the specification of systems comprising multiple components, referred to as modules, that operate in parallel, both asynchronously and synchronously through action labels. Each module has a number of finite-valued variables and a state of the system specifies the values of the variables of all modules. The behaviour of each module is defined by probabilistic guarded commands, where the guard is a predicate over the variables of the modules and the command specifies a probabilistic update of the module's variables. In a CSG model, each player constitutes a set of modules, and these therefore execute concurrently.

PRISM-games provides a graphical user interface for designing and simulating stochastic games models, but its core functionality is to exhaustively construct a game and perform verification and strategy synthesis against a logical specification. For CSGs, the PRISM-games logic described in Definition 13 is supported, and the resulting strategies can be exported, simulated or further verified.

The implementation of CSG model checking is built within PRISM's 'explicit' engine, which is based on sparse matrices and implemented in Java. Computing values (and optimal strategies) for zero-sum NFGs, needed for zero-sum formulae, is performed using the LPSolve library [41] via linear programming. The computation of SWNE or SFNE for nonzero-sum NFG, required for nonzero-sum formulae, depends on the number of players. For two players [34], labelled polytopes are used to characterise and find NE values through a reduction to SMT in both Z3 [19] and Yices [20]. If there are more than two players, the implementation [32] is based on support enumeration and uses a combination of the SMT solver Z3 [19] and the nonlinear optimisation suite Ipopt [58]. In the case of SWCE for nonzero-sum NFGs, as the problem reduces to an LP problem [35], either Gurobi [22] or the SMT solver Z3 [19] is used. Finally, for SFCE, since the problem does not reduce directly to an LP problem, only Z3 can be used.

5.2 The ALOHA case study

We now return to the slotted ALOHA protocol discussed in Examples 12 and 14 to illustrate the benefits of game-theoretic analysis with CSGs. For further details of this case study, as well as several others, see [61]. Recall that, in the slotted ALOHA protocol, a number of users are attempting to send packets on a shared medium. We assume that, in any time slot, if a single user tries to send a packet then there is a probability (q) that the packet is sent and, as more users try and send, then the probability of success decreases. If sending a packet fails, the user waits for a number of slots before resending, defined 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 $\{0, 1, \ldots, 2^k{-}1\}$.

Zero-sum properties. We first consider the zero-sum properties $\langle\!\langle usr_1 \rangle\!\rangle \mathtt{P}_{\max=?}[\,\mathtt{F}^{\leqslant D} \mathsf{sent}_1\,]$ and $\langle\!\langle usr_1 \rangle\!\rangle \mathtt{R}^{time}_{\min=?}[\,\mathtt{F}\ \mathsf{sent}_1\,]$ from Example 14, which correspond to the first user trying to maximise the probability that their packet is sent before a deadline and trying to minimise the expected time to send their packet, respectively. The results for the first property when $q = 0.9$ as the deadline D varies, and for the second property as the probability q varies, are presented in Figure 2 for different values of $b_{\max}$. We see that the probability decreases and the expected time decreases as $b_{\max}$ increases; this is because, as $b_{\max}$ increases, the additional time the first user can spend in backoff outweighs the gains in reducing the chance of avoiding further collisions. By performing strategy synthesis we see that it is optimal for the first user to initially randomly decide as to when to send their packet in order to avoid collisions with the coalition of the second and third user. However, this changes to a deterministic strategy of just sending its packet when the other users have sent their packets or the deadline is getting close, and therefore waiting will mean the deadline is missed.

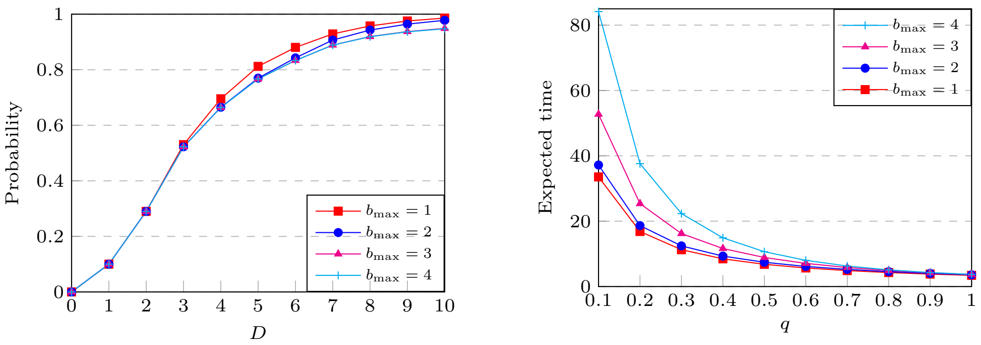


Figure 2 Results from a CSG model of the ALOHA protocol: one user maximising the probability of sending their packet before a deadline D (left); and minimising the expected time to send the packet, assuming a message transmission failure probability q (right).

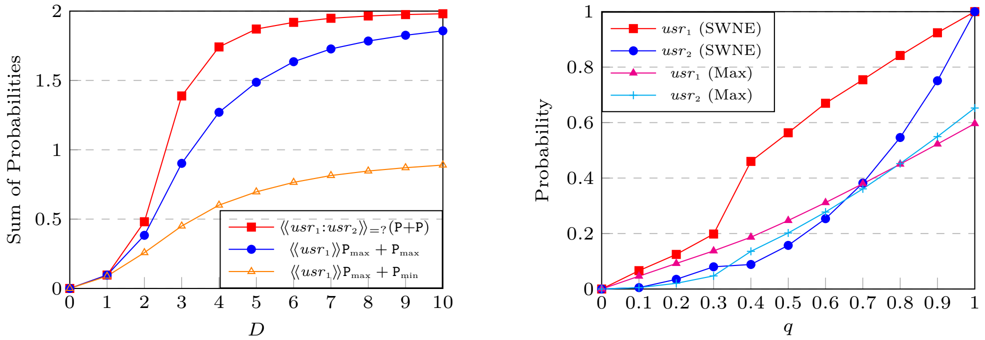


Figure 3 Results from CSG equilibria synthesis on the ALOHA protocol, maximising the probabilities of sending packets by deadline D for two coalitions (user 1, and users 2 and 3): probability sums (left) and individual probabilities (right).

Benefits of equilibria. We next highlight the analysis from [34], which demonstrates the advantages of cooperation through nonzero-sum properties when using Nash equilibria (NE) and the social welfare (SW) optimality criterion, as opposed to adopting a strategy that assumes antagonistic behaviour. The first non-zero sum property we consider corresponds to the case when each user is trying to maximise the probability of sending their packet before a deadline D, with users 2 and 3 forming a coalition, represented by the formula $\langle\langle usr_1{:}usr_2{,}usr_3\rangle\rangle(\mathrm{NE},\mathrm{SW})_{\max=?}(\mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_1 \wedge t{\leqslant}D)\,] + \mathtt{P}[\,\mathtt{F}\ (\mathsf{sent}_2 \wedge \mathsf{sent}_3 \wedge t{\leqslant}D)\,])$.

Figure 3 presents total values (the sum of the probabilities for user 1 and the coalition of user 2 and 3) as D varies (left) and individual values as q varies (right). By performing strategy synthesis, the analysis found that the collaboration is dependent on both D and q. In particular, if the users have more time there is a greater chance for the users to collaborate by sending in different slots, whereas, when q is large, it is unlikely users need to repeatedly send, so again can send in different slots. As Figure 3 (right) demonstrates, since the coalition has more packets to send, their probabilities are lower.

Equilibria types and optimality criteria. Finally, we report on the experiments of [35], which investigate the benefits of using different types of equilibria, i.e., *correlated* (CE) over *Nash* equilibria, and optimality criteria, i.e., *social fairness* (SF) over *social welfare* (SW). The experiments varied the number of users and considered the case when the objective of each individual user is to minimise the expected time to send their packet, which is represented by the nonzero-sum formula $\langle\langle usr_1{:}\cdots{:}usr_m\rangle\rangle(\star_1,\star_2)_{\min=?}(\mathtt{R}^{time}[\,\mathtt{F}\ \mathsf{sent}_1\,]{+}\cdots{+}\mathtt{R}^{time}[\,\mathtt{F}\ \mathsf{sent}_m\,])$.

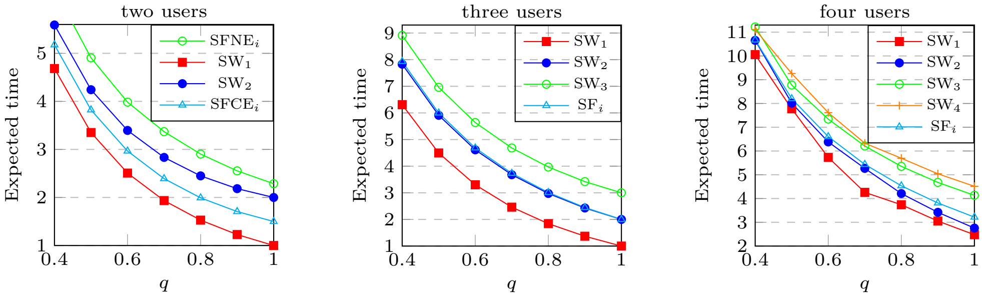


Figure 4 Results from different types of equilibria (correlated vs. Nash) and optimality criteria (social fairness vs. social welfare) for minimising the expected times for users to send packets in the ALOHA protocol., for varying numbers of users.

Synthesising optimal strategies for this specification, it was found that the cases for SWNE and SWCE coincide (although SWCE returns a joint strategy for the users, this joint strategy can be separated to form a strategy profile). This profile required one user to try and send first, and then for the remaining users to take turns to try and send afterwards. If a user fails to send, then they enter backoff and allow all remaining users to try and send before trying to send again. The reason for this is that there is no gain in a user trying to send at the same time as another user, as this will increase the probability of a collision and thus their packets not being sent, and therefore the users having to spend time in backoff.

For SFNE, which has only been implemented for the two-player case, the two users followed identical strategies, which involve randomly deciding whether to wait or transmit, unless they are the only user that has not transmitted, and then they always try to send when not in backoff. In the case of SFCE, users employed a shared probabilistic signal to coordinate which user sends next. Initially, this was a uniform choice over the users, but as time progresses the signal favoured the users with lower backoff counters as these users had fewer opportunities to send their packet previously.

Figure 4 plots the optimal values for the users, where SW$_i$ corresponds to the optimal values (expected times to send their packets) for user i for both SWNE and SWCE for the cases of two, three and four users. We see that the optimal values for the different users under SFNE and SFCE coincide, while under SWNE and SWCE they are different for each user (with the user sending first having the lowest and the user sending last the highest). Comparing the sum of the SWNE (and SWCE) values and that of the SFCE values, we see a small decrease in the sum of less than 2% of the total, whereas for SFNE there is a greater difference as the users cannot coordinate, and hence try and send at the same time.

6 Recent Developments: Neuro-symbolic CSGs

The recent encouraging advances of AI, and particularly deep learning, have resulted in computing architectures that integrate components that are synthesized from data (e.g., implemented as neural networks) with conventional, symbolic modules (e.g., controllers). Design automation support for such *neuro-symbolic* systems is, however, lacking. To this end, we have developed the model of *neuro-symbolic concurrent stochastic games* (NS-CSGs) [60, 59], which is targeted at AI-based autonomous systems, e.g., autonomous driving or aircraft controllers. NS-CSGs are a variant of (continuous-space) CSGs, in which each player is a neuro-symbolic *agent* and the agents act concurrently in a shared, continuous-state environment. As for the players of CSGs, each agent has a finite set of available actions and agents choose their actions simultaneously; however, in NS-CSGs the action choices cause the agents' local states to be updated probabilistically and the agents are endowed with a perception mechanism implemented as a neural network, through which they can observe the local states of the other agents and that of the environment and encode these observations as locally stored *percepts*. The global states of NS-CSGs comprise the state of the environment together with the local state and percept of each agent, and are therefore infinite-state, in contrast to the CSGs discussed in the rest of this paper.

▶ **Definition 15** (Neuro-symbolic concurrent stochastic game [60, 59])**.** *A neuro-symbolic concurrent stochastic multi-player game (NS-CSG)* NSC *comprises players* $(\mathsf{Ag}_i)_{i\in N}$*, for* $N = \{1,\dots,n\}$*, and an environment* E *where:*

$$\mathsf{Ag}_i = (S_i, A_i, \Delta_i, obs_i, \delta_i) \text{ for } i \in N, \quad E = (S_E, \delta_E)$$

and we have:

- $S_i = Loc_i \times Per_i$ *is a set of states for* Ag_i*, where* $Loc_i \subseteq \mathbb{R}^{b_i}$ *and* $Per_i \subseteq \mathbb{R}^{d_i}$ *are finite sets of local states and percepts, respectively;*
- $S_E \subseteq \mathbb{R}^e$ *is a closed infinite set of environment states;*
- A_i *is a nonempty finite set of actions for* Ag_i *and* $A := (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$ *is the set of* joint actions*, where* $\bot$ *is an idle action disjoint from* $\cup_{i=1}^n A_i$*;*
- $\Delta_i : S_i \to 2^{A_i}$ *is an available action function, defining the actions* Ag_i *can take in each state;*
- $obs_i : (Loc_1 \times \cdots \times Loc_n \times S_E) \to Per_i$ *is a perception function for* Ag_i*, mapping the local states of all agents and the environment to a percept of the agent, implemented via a neural network (NN) classifier;*
- $\delta_i : S_i \times A \to \mathbb{P}(Loc_i)$ *is a partial probabilistic transition function for* Ag_i *determining the distribution over the agent's local states given its current state and joint action;*
- $\delta_E : S_E \times A \to S_E$ *is a partial deterministic environment transition function determining the environment's next state given its current state and joint action.*

A (global) state for an NS-CSG NSC comprises a state $s_i = (loc_i, per_i)$ for each agent Ag_i (a pair of a local state and percept) and an environment state s_E. If an NS-CSG is in a state $s = (s_1, \dots, s_n, s_E)$, then each Ag_i simultaneously chooses one of the actions available in its state s_i (if no action is available, i.e., $\Delta_i(s_i) = \varnothing$, it picks the idle action $\bot$) yielding a joint action $\alpha = (a_1, \dots, a_n) \in A$. Next, each Ag_i updates its local state to $loc'_i \in Loc_i$, according to its probabilistic local transition function δ_i, applied to its current state (loc_i, per_i) and the joint action α. The environment updates its state to $s'_E \in S_E$ according to its local deterministic transition function δ_E, based on its state s_E and on α. Finally, each agent, based on its new local state loc'_i, observes the new local states of the other agents and environment through its perception function obs_i to generate a new percept $per'_i = obs_i(loc'_1, \dots, loc'_n, s'_E)$. Thus, the game reaches the state $s' = (s'_1, \dots, s'_n, s'_E)$, where $s'_i = (loc'_i, per'_i)$ for $i \in N$. We assume that each perception function obs_i is implemented via an NN $f_i : \mathbb{R}^{b+e} \to \mathbb{P}(Per_i)$ yielding a normalised score over different percept values, where $b = \sum_{i=1}^n b_i$; however, it can be any function including other types of machine learning models. A rule is then applied that selects the percept value with the maximum score.

Formally, the semantics of an NS-CSG NSC is given by an infinite-state CSG $[\![\mathsf{NSC}]\!]$ over the product of the states of the agents and environment, which assumes a particular structure of the transition function that distinguishes between agent and environment states and uses the NN-based perception function to define which states have the same characteristics.

▶ **Definition 16** (Semantics of an NS-CSG)**.** *Given an NS-CSG* NSC *consisting of* n *players and an environment, the semantics of* NSC *is* $[\![\mathsf{NSC}]\!] = (N, S, A, \Delta, \delta)$ *where:*

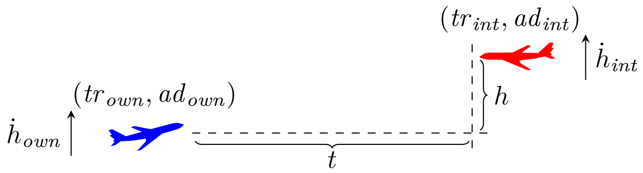


Figure 5 Geometry with trust levels and advisories for the agents of the VCAS[2] case study.

- $S = S_1 \times \cdots \times S_n \times S_E$ *is the set of (global) states, which contain both discrete and continuous elements;*
- $A = (A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\})$*;*
- $\Delta(s_1, \ldots, s_n, s_E) = \cup_{i=1}^{n} \Delta_i(s_i)$*;*
- $\delta : (S \times ((A_1 \cup \{\bot\}) \times \cdots \times (A_n \cup \{\bot\}))) \to \mathbb{P}(S)$ *is the partial probabilistic transition function, where for states* $s = (s_1, \ldots, s_n, s_E), s' = (s'_1, \ldots, s'_n, s'_E) \in S$ *and joint action* $\alpha = (a_1, \ldots, a_n) \in A$*, if* $a_i \in \Delta_i(s_i)$ *when* $\Delta_i(s_i) \neq \varnothing$ *and* $a_i = \bot$ *otherwise, then* $\delta(s, \alpha)$ *is defined and if* $s'_i = (loc'_i, per'_i)$*,* $per'_i = obs_i(loc'_1, \ldots, loc'_n, s'_E)$ *for all* $i \in N$ *and* $s'_E = \delta_E(s_E, \alpha)$*, then*

  $$\delta(s, \alpha)(s') = \left(\prod_{i=1}^{n} \delta_i(s_i, \alpha)(loc'_i)\right)$$

  and otherwise $\delta(s, \alpha)(s') = 0$*.*

To illustrate NS-CSGs, we present the VerticalCAS Collision Avoidance Scenario (VCAS[2]) [28, 29], modelled in [59], which is a variant of the one studied in [2], where the agents' trust level is modelled probabilistically to account for possible uncertainty.

▶ **Example 17.** The geometry of the VCAS[2] case study is shown in Figure 5. There are two aircraft (ownship and intruder), constituting the agents of the NS-CSG, both equipped with an NN-controlled collision avoidance system called VCAS. At each time unit (i.e., every second), VCAS issues an advisory ($ad \in \{1, \ldots, 9\}$) from which, together with the current trust level ($tr \in \{1, \ldots, 4\}$) from the previous advisory, the pilot needs to make a decision about the rate of acceleration, aimed at avoiding a near mid-air collision (NMAC) [1].

The input to the VCAS system is the tuple $(h, \dot{h}_{own}, \dot{h}_{int}, t) \in \mathbb{R}^4$ including the relative altitude h of the aircraft, the climb rate $\dot{h}_{own}$ of ownship, the climb rate $\dot{h}_{int}$ of intruder, and the time t until loss of horizontal separation between the aircraft. VCAS is implemented via nine feed-forward NNs $F = \{f_i : \mathbb{R}^4 \to \mathbb{R}^9 \mid 1 \leqslant i \leqslant 9\}$, each of which corresponds to an advisory and outputs the scores of the nine possible advisories. Each advisory will provide a set of accelerations for the agent to select from and the trust level increases probabilistically if the current advisory is compliant with the executed accelerations, and decreases otherwise. We formulate the NS-CSG with agents Ag_i for $i \in \{own, int\}$ as follows:

- the set of states for Ag_i is given by $S_i = \{1, \ldots, 4\} \times \{1, \ldots, 9\}$, where the agent state $s_i = (tr_i, ad_i) \in S_i$ has local state (trust level) tr_i and percept (advisory) ad_i;
- the set of environment states is given by $S_E = \mathbb{R}^4$, where $s_E = (h, \dot{h}_{own}, \dot{h}_{int}, t) \in S_E$ represents the relative altitude, the climb rate of the ownship, the climb rate of the intruder, and the time until loss of their horizontal separation;
- the set of actions of Ag_i is given by $A_i = \{0, \pm 3.0, \pm 7.33, \pm 9.33, \pm 9.7, \pm 11.7\}$ representing the acceleration options of Ag_i;
- the available action function $\Delta_i : Per_i \to A_i$ of Ag_i is independent of the local state of the agent and returns the set consisting two non-zero acceleration actions from Table 2 for a given percept and the zero acceleration action;

Label (ad_i)	Advisory	Description	Vertical Range (Min, Max) ft/min	Actions ft/s^2
1	COC	Clear of Conflict	$(-\infty, +\infty)$	-3, $+3$
2	DNC	Do Not Climb	$(-\infty, 0]$	-9.33, -7.33
3	DND	Do Not Descend	$[0, +\infty)$	7.33, $+9.33$
4	DES1500	Descend at least 1500 ft/min	$(-\infty, -1500]$	-9.33, -7.33
5	CL1500	Climb at least 1500 ft/min	$[+1500, +\infty)$	$+7.33$, $+9.33$
6	SDES1500	Strengthen Descend to at least 1500 ft/min	$(-\infty, -1500]$	-11.7, -9.7
7	SCL1500	Strengthen Climb to at least 1500 ft/min	$[+1500, +\infty)$	$+9.7$, $+11.7$
8	SDES2500	Strengthen Descend to at least 2500 ft/min	$(-\infty, -2500]$	-11.7, -9.7
9	SCL2500	Strengthen Climb to at least 2500 ft/min	$[+2500, +\infty)$	$+9.7$, $+11.7$

Table 2 Actions available for the agents of VCAS[2] for each advisory [2].

- the perception function $obs_i : Per_i \times S_E \to \{1, \ldots, 9\}$ of Ag_i is independent of the local state of the agent and is given by the feed forward NNs F of VCAS;
- the local transition function δ_i of Ag_i updates the agent's trust level probabilistically according to its current trust level, its current advisory and its executed action;
- the environment transition function δ_E is given by $\delta_E((h, \dot{h}_{own}, \dot{h}_{int}, t), (\ddot{h}_{own}, \ddot{h}_{int})) = (h', \dot{h}'_{own}, \dot{h}'_{int}, t')$ where for time step $\Delta t = 1$:
  - $h' = h - \Delta t(\dot{h}_{own} - \dot{h}_{int}) - 0.5\Delta t^2(\ddot{h}_{own} - \ddot{h}_{int})$;
  - $\dot{h}'_{own} = \dot{h}_{own} + \ddot{h}_{own}\Delta t$;
  - $\dot{h}'_{int} = \dot{h}_{int} + \ddot{h}_{int}\Delta t$;
  - $t' = t - \Delta t$.

6.1 Zero-sum NS-CSGs

In view of the uncountable state spaces, [60] presents an approach for zero-sum *discounted infinite-horizon* cumulative rewards under the assumption of full state observability for NS-CSGs, which exploits Borel state space decomposition and identifies model restrictions to ensure determinacy, and therefore existence of a value that corresponds to a unique fixed point. Value iteration and policy iteration algorithms to compute values and synthesise optimal strategies are also derived based on formulating piecewise linear or constant representations of the value functions and strategies for NS-CSGs.

6.2 Nonzero-sum NS-CSGs

In the case of nonzero-sum NS-CSGs, [59] studies the *undiscounted*, *finite-horizon* equilibria synthesis problem. The use of finite-horizon objectives simplifies the analysis (note that the existence of infinite-horizon NE for CSGs is an open problem [7], and the verification of non-probabilistic infinite-horizon reachability properties for neuro-symbolic games is undecidable [2]). Both NE and CE using the SW optimality criteria are considered. The algorithms, based on backward induction and non-linear programming, compute *globally optimal* equilibria which, from a fixed initial state, are optimal over the chosen time horizon, in contrast to the local optimality of equilibria for finite-state CSGs [34, 35].

▶ **Example 18.** The NS-CSG model of the VCAS[2] system described in Example 17 is studied in [59], comparing equilibria strategies to the zero-sum strategies analysed in [2]. Figure 6 plots the relative altitude h of the two aircraft for equilibria and zero-sum strategies when maximising this value for a given time instant k, plotted for several different initial

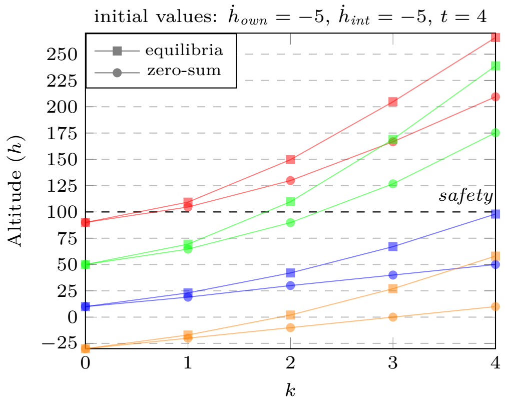


Figure 6 Relative altitude h of the two aircraft at time instants k for equilibria and zero-sum strategies for the VCAS[2] case study.

values of h. It can be seen that, with respect to the safety criterion established in [28, 2], i.e., avoiding an NMAC when two aircraft are separated by less than 100 ft vertically (dotted line) and 500 ft horizontally, equilibria strategies allow the two aircraft to reach a safe configuration within a shorter horizon, which would be missed through a zero-sum analysis.

The analysis of [59] also considers a reward structure that incorporates both the trust level and fuel consumption. Figure 7 shows the resulting equilibria strategies when both safety and trust are prioritised, using two different time horizons. When $t = 3$ initially (left), it is optimal to follow the advisories and the trust levels tr_{own} and tr_{int} of the two aircraft never decrease from their initial values of 4. However, when $t = 3$ initially (right), the optimal strategy shows a deviation from the advisory, denoted by action $a_{own} = 0$, in state s_2, resulting in tr_{own} dropping to 3 in s_3 with probability 0.9.

The experiments from [59], summarised above, and from [60] are developed using prototype tools that build upon parts of PRISM-games, but full tool support for NS-CSGs is not yet provided. Efforts in this direction are continuing.

7 Conclusions and Future Challenges

This paper has provided an overview of modelling, verification and strategy synthesis techniques that have been developed and implemented for concurrent stochastic games in the PRISM-games model checker. Through case studies, we have demonstrated the uses and advantages of zero-sum and equilibria-based reasoning in strategic decision making, highlighting Nash and correlated equlilibria in conjunction with two optimality criteria, social welfare and social fairness. We have also discussed recent trends in autonomous systems towards neural-symbolic architectures, and summarised the first steps towards developing a modelling framework to support the development of such AI-based systems. Despite some progress, many problems remain open in this area, in particular, the development of (efficient) approximate algorithms for (undiscounted infinite-horizon) temporal logic specifications where the underlying problem is undecidable even in the finite-state case [42].

References

1 M. Akintunde, E. Botoeva, P. Kouvaros, and A. Lomuscio. Formal verification of neural agents in non-deterministic environments. In *Proc. AAMAS'20*, pages 25–33. Springer, 2020.

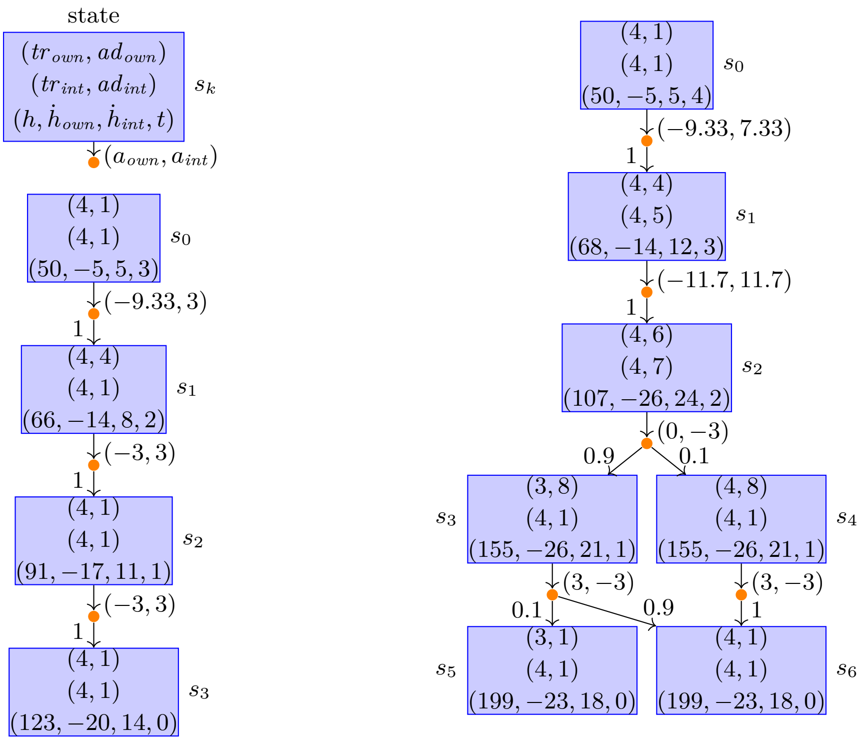


Figure 7 Optimal strategies for the VCAS[2] case study over two different time horizons, using initial t values of 3 (left) and 4 (right).

2 M. Akintunde, E. Botoeva, P. Kouvaros, and A. Lomuscio. Verifying Strategic Abilities of Neural-symbolic Multi-agent Systems. In *Proc. KR'20*, pages 22–32. IJCAI Organization, 9 2020.

3 R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

4 R. Alur, T. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

5 R. Alur, T. Henzinger, F. Mang, S. Qadeer, S. Rajamani, and S. Tasiran. MOCHA: Modularity in model checking. In *Proc. 10th Int. Conf. Computer Aided Verification (CAV'98)*, volume 1427 of *LNCS*, pages 521–525, Vancouver, 1998. Springer.

6 R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.

7 P. Bouyer, N. Markey, and D. Stan. Mixed Nash equilibria in concurrent games. In *Proc. FSTTCS'14*, volume 29 of *LIPICS*, pages 351–363, 2014.

8 R. Brenguier. PRALINE: A tool for computing Nash equilibria in concurrent games. In *Proc. CAV'13*, volume 8044 of *LNCS*, pages 890–895. Springer, 2013. lsv.fr/Software/praline/.

9 T. Brihaye, V. Bruyère, A. Goeminne, J.-F. Raskin, and M. van den Bogaard. The complexity of subgame perfect equilibria in quantitative reachability games. In *Proc. CONCUR'19*, volume 140 of *LIPIcs*, pages 13:1–13:16. Leibniz-Zentrum für Informatik, 2019.

10 P. Čermák, A. Lomuscio, F. Mogavero, and A. Murano. MCMAS-SLK: A model checker for the verification of strategy logic specifications. In *Proc. CAV'14*, volume 8559 of *LNCS*, pages 525–532. Springer, 2014.

11 K. Chatterjee, L. de Alfaro, and T. Henzinger. Strategy improvement for concurrent reachability and turn-based stochastic safety games. *Journal of Computer and System Sciences*, 79(5):640–657, 2013.

12 K. Chatterjee and T. Henzinger. Value iteration. In *25 Years of Model Checking*, volume 5000 of *LNCS*, pages 107–138. Springer, 2008.

13 K. Chatterjee, T. Henzinger, B. Jobstmann, and A. Radhakrishna. Gist: A solver for probabilistic games. In *Proc. CAV'10*, volume 6174 of *LNCS*, pages 665–669. Springer, 2010. pub.ist.ac.at/gist/.

14 K. Chatterjee, R. Majumdar, and M. Jurdziński. On Nash equilibria in stochastic games. In *Proc. CSL'04*, volume 3210 of *LNCS*, pages 26–40. Springer, 2004.

15 T. Chen, V. Forejt, M. Kwiatkowska, D. Parker, and A. Simaitis. Automatic verification of competitive stochastic systems. *Formal Methods in System Design*, 43(1):61–92, 2013.

16 C. Cheng, A. Knoll, M. Luttenberger, and C. Buckl. GAVS+: An open platform for the research of algorithmic game solving. In *Proc. TACAS'11*, volume 6605 of *LNCS*, pages 258–261. Springer, 2011. sourceforge.net/projects/gavsplus/.

17 L. de Alfaro, T. Henzinger, and O. Kupferman. Concurrent reachability games. *Theoretical Computer Science*, 386(3):188–217, 2007.

18 L. de Alfaro and R. Majumdar. Quantitative solution of omega-regular games. *Journal of Computer and System Sciences*, 68(2):374–397, 2004.

19 L. De Moura and N. Bjørner. Z3: An efficient SMT solver. In *Proc. TACAS'08*, volume 4963 of *LNCS*, pages 337–340. Springer, 2008. github.com/Z3Prover/z3.

20 B. Dutertre. Yices 2.2. In *Proc. CAV'14*, volume 8559 of *LNCS*, pages 737–744. Springer, 2014. yices.csl.sri.com.

21 V. Forejt, M. Kwiatkowska, G. Norman, and D. Parker. Automated verification techniques for probabilistic systems. In *SFM'11*, volume 6659 of *LNCS*, pages 53–113. Springer, 2011.

22 Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2021. gurobi.com.

23 J. Gutierrez, M. Najib, P. Giuseppe, and M. Wooldridge. Equilibrium design for concurrent games. In *Proc. CONCUR'19*, volume 140 of *LIPIcs*, pages 22:1–22:16. Leibniz-Zentrum für Informatik, 2019.

24 J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. EVE: A tool for temporal equilibrium analysis. In *Proc. ATVA'18*, volume 11138 of *LNCS*, pages 551–557. Springer, 2018. github.com/eve-mas/eve-parity.

25 Julian Gutierrez, Lewis Hammond, Anthony W. Lin, Muhammad Najib, and Michael J. Wooldridge. Rational verification for probabilistic systems. In *Proc. 18th International Conference on Principles of Knowledge Representation and Reasoning (KR'21)*, pages 312–322, 2021.

26 H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *FAC*, 6(5):512–535, 1994.

27 L. Hurwicz and S. Reiter. *Designing Economic Mechanisms.* Cambridge University Press, 2006.

28 K. Julian and M. Kochenderfer. A reachability method for verifying dynamical systems with deep neural network controllers. *CoRR*, abs/1903.00520, 2019.

29 K. Julian, S. Sharma, J.-B. Jeannin, and M. Kochenderfer. Verifying aircraft collision avoidance neural networks through linear approximations of safe regions. *CoRR*, abs/1903.00762, 2019.

30 N. Karmarkar. A new polynomial-time algorithm for linear programming. *Combinatorica*, 4(4):373–395, 1984.

31 J. Kemeny, J. Snell, and A. Knapp. *Denumerable Markov Chains.* Springer, 1976.

32 M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. Multi-player equilibria verification for concurrent stochastic games. In *Proc. QEST'20*, volume 12289 of *LNCS*, pages 74–95. Springer, 2020.

33 M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. PRISM-games 3.0: Stochastic game verification with concurrency, equilibria and time. In *Proc. CAV'20*, volume 12225 of *LNCS*, pages 475–487. Springer, 2020.

34 M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. Automatic verification of concurrent stochastic systems. *Formal Methods in System Design*, pages 1–63, 2021.

35 M. Kwiatkowska, G. Norman, D. Parker, and G. Santos. Correlated equilibria and fairness in concurrent stochastic games. In *Proc. TACAS'22*, volume 13244 of *LNCS*, pages 60–78. Springer, 2022.

36 S. LaValle. Robot motion planning: A game-theoretic foundation. *Algorithmica*, 26:430–465, 2000.

37 C. Lemke and Jr J. Howson. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423, 1964.

38 M. Littman, N. Ravi, A. Talwar, and M. Zinkevich. An efficient optimal-equilibrium algorithm for two-player game trees. In *Proc. UAI'06*, pages 298–305. AUAI Press, 2006.

39 A. Lomuscio, H. Qu, and F. Raimondi. MCMAS: A model checker for the verification of multi-agent systems. In *Proc. 21st Int. Conf. Computer Aided Verification (CAV'09)*, volume 5643 of *LNCS*, pages 682–688. Springer, 2009.

40 D. Lozovanu and S. Pickl. Determining Nash equilibria for stochastic positional games with discounted payoffs. In *Proc. ADT'17*, volume 10576 of *LNAI*, pages 339–343. Springer, 2017.

41 LPSolve (version 5.5). lpsolve.sourceforge.net/5.5/.

42 O. Madani, S. Hanks, and A. Condon. On the undecidability of probabilistic planning and related stochastic optimization problems. *Artificial Intelligence*, 147(1–2):5–34, 2003.

43 J. Marden and J. Shamma. Game theory and control. *Annual Review of Control, Robotics, and Autonomous Systems*, 1(1):105–134, 2018.

44 D. Martin. The determinacy of Blackwell games. *J. Symbolic Logic*, 63(4):1565–1581, 1998.

45 R. McKelvey, A. McLennan, and T. Turocy. Gambit: Software tools for game theory. gambit-project.org.

46 N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007.

47 M. Osborne and A. Rubinstein. *An Introduction to Game Theory*. Oxford University Press, 2004.

48 R. Porter, E. Nudelman, and Y. Shoham. Simple search methods for finding a Nash equilibrium. In *Proc. AAAI'04*, pages 664–669. AAAI Press, 2004.

49 H. Prasad, L. Prashanth, and S. Bhatnagar. Two-timescale algorithms for learning Nash equilibria in general-sum stochastic games. In *Proc. AAMAS'15*, pages 1371–1379. IFAAMAS, 2015.

50 E. Prisner. *Game Theory Through Examples*. Mathematical Association of America, 1 edition, 2014.

51 T. Raghavan and J. Filar. Algorithms for stochastic games — a survey. *Zeitschrift für Operations Research*, 35(6):437–472, Nov 1991.

52 T. Sandholm, A. Gilpin, and V. Conitzer. Mixed-integer programming methods for finding Nash equilibria. In *Proc. AAAI'05*, pages 495–501. AAAI Press, 2005.

53 U. Schwalbe and P. Walker. Zermelo and the early history of game theory. *Games and Economic Behavior*, 34(1):123–137, 2001.

54 L. Shapley. Stochastic games. *PNAS*, 39:1095–1100, 1953.

55 A. Toumi, J. Gutierrez, and M. Wooldridge. A tool for the automated verification of Nash equilibria in concurrent games. In *Proc. ICTAC'15*, volume 9399 of *LNCS*, pages 583–594. Springer, 2015.

56 J. von Neumann. Zur theorie der gesellschaftsspiele. *Mathematische Annalen*, 100:295–320, 1928.

57 J. von Neumann, O. Morgenstern, H. Kuhn, and A. Rubinstein. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

58 A. Wächter. Short tutorial: Getting started with IPOPT in 90 minutes. In *Combinatorial Scientific Computing*, number 09061 in Dagstuhl Seminar Proceedings. Leibniz-Zentrum für Informatik, 2009. github.com/coin-or/Ipopt.

59 R. Yan, G. Santos, X. Duan, D. Parker, and M. Kwiatkowska. Finite-horizon equilibria for neuro-symbolic concurrent stochastic games. In *Proc. UAI'22*, 2022.

60 R. Yan, G. Santos, G. Norman, D. Parker, and M. Kwiatkowska. Strategy synthesis for zero-sum neuro-symbolic concurrent stochastic games. arXiv.2202.06255, 2022.

61 PRISM-games web site. prismmodelchecker.org/games/.