Mean-Payoff Games with ω-Regular Specifications

Thomas Steeples
Department of Computer Science
University of Oxford
Oxford, United Kingdom
thomas.steeples@cs.ox.ac.uk

Julian Gutierrez
Faculty of Information Technology
Monash University
Melbourne, Australia
julian.gutierrez@monash.edu

Michael Wooldridge
Department of Computer Science
University of Oxford
Oxford, United Kingdom
michael.wooldridge@cs.ox.ac.uk

ABSTRACT

Multi-player mean-payoff games are a natural formalism to model concurrent and multi-agent systems with self-interested players. Players in such a game traverse a graph, while trying to maximise a mean-payoff function that depends on the plays so generated. As with all games, the equilibria that could arise may have undesirable properties. However, as system designers, we typically wish to ensure that equilibria in such systems correspond to desirable system behaviours, for example, satisfying certain safety or liveness properties. One natural way to do this would be to specify such desirable properties using temporal logic. Unfortunately, the use of temporal logic specifications causes game theoretic verification problems to have very high computational complexity. To this end, we consider ω-regular specifications, which offer a concise and intuitive way of specifying desirable behaviours of a system. The main results of this work are characterisation and complexity bounds for the problem of determining if there are equilibria that satisfy a given ω-regular specification in a multi-player mean-payoff game in a number of computationally relevant game-theoretic settings.

CCS CONCEPTS

• **Theory of computation** → Logic & verification; • **Computing methodologies** → Multi-agent systems.

KEYWORDS

Multi-player games, Mean-payoff games, Automated verification, Temporal logic, Game theory, Equilibria, Multi-agent systems

ACM Reference Format:

Thomas Steeples, Julian Gutierrez, and Michael Wooldridge. 2021. Mean-Payoff Games with ω-Regular Specifications. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Modelling concurrent and multi-agent systems as games in which players interact by taking actions in pursuit of their preferences is an increasingly common approach in both formal verification and artificial intelligence [1, 3, 29]. One widely adopted semantic framework for modelling such systems is that of concurrent game structures [3]. Such structures capture the dynamics of a system — the actions that agents/players can perform, and the effects of these actions. On top of this framework, we can impose additional structure to represent each player's preferences over the possible runs of the system. There are several extant approaches to capturing preferences. One natural quantitative method involves assigning a weight to every state of the game, and then considering each player's *mean-payoff* over generated runs: a player prefers runs that maximise their mean-payoff [11, 41, 45]. These games are effective in modelling resource-bounded reactive systems, as well as any scenario with multiple agents and quantitative features. Under the assumption that each agent in the system is acting rationally, concepts from game theory offer a natural framework for understanding its possible behaviours [35]. This approach is expressive enough to capture many applications of interest, and has been receiving increasing attention recently [9]. As such, equilibria for multi-player games with mean-payoff objectives are well studied, and the computation of Nash equilibria in such games has been shown to be NP-complete over state-transition graphs [41].

However, a given game-theoretic equilibrium may have undesirable computational properties from the point of view of a system designer. An equilibrium may visit dangerous states, or get stuck in a deadlock. Thus, one may also want to check if there exist equilibria which satisfy some additional desirable properties associated with the game. This problem – determining whether a given formal specification is satisfied on some/every equilibrium of a multi-agent system – is known as *Rational Verification* [12, 44].

Previous approaches to rational verification have borrowed their methodology from temporal logic verification; cf., [13, 17–19, 26]. However, since rational verification subsumes automated synthesis, the use of temporal logic specifications introduces high computational complexity [37]. To mitigate this problem, one might use fragments of temporal logic with lower complexity (*e.g.,* GR(1) [7, 25]), but in this work we adopt a different approach. Taking inspiration from automata theory, and in particular from [6], we consider system specifications given by a formal language for expressing *ω-regular specifications*, defined in terms of those states in the system that are visited infinitely often. With this approach, the complexity of the main game-theoretic decision problems is considerably lower than in the case with temporal logic specifications.

In this paper, we offer the following main contributions: we introduce a syntax for ω-regular specifications and demonstrate they are a natural construct for reasoning qualitatively about concurrent games. We then study multi-player mean-payoff games with ω-regular specifications in the non-cooperative setting [35], and consider the natural decision problems relating to these games and their Nash equilibria. Following this, we take inspiration from cooperative game theory and look at equivalent decision problems with respect to a cooperative solution concept derived from the core [20, 35]. Finally, we look at reactive module games [18] as a way of inducing succinctness in our system representations, and look at how this affects our established complexity results.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.

Structure of the paper. After introducing some necessary background, we give a motivating example, define the main game-theoretic framework, and discuss some of its properties in Section 2. In Sections 3, 4 and 5, we present the main results, and in Section 6 we discuss some relevant related work.

2 MODELS, GAMES, AND SPECIFICATIONS

Games. A concurrent game structure [3] is a tuple,

$$M = (\mathrm{Ag}, \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i \in \mathrm{Ag}}, \mathrm{tr}),$$

where,

- Ag and St are finite, non-empty sets of agents and system states, respectively, where $s^0 \in \mathrm{St}$ is an initial state;
- Ac_i is a set of actions available to agent i, for each i;
- $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to \mathrm{St}$ is a transition function.

We define the *size* of M to be $|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$.

Concurrent games are played as follows. The game begins in state s^0, and each player $i \in \mathrm{Ag}$ simultaneously picks an action $\mathrm{ac}_i^0 \in \mathrm{Ac}_i$. The game then transitions to a new state, $s^1 = \mathrm{tr}(s^0, \mathrm{ac}_1^0, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^0)$, and this process repeats. Thus, the n^{th} state visited is $s^n = \mathrm{tr}(s^{n-1}, \mathrm{ac}_1^{n-1}, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^{n-1})$. Since the transition function is deterministic, a play of a game will be an infinite sequence of states, $\pi : \mathbb{N} \to \mathrm{St}$. We call such a sequence of states a *run*. Typically, we index runs with square brackets, *i.e.*, the kth state visited in the run π is denoted $\pi[k]$, and we also use *slice* notation to denote prefix, suffixes and fragments of runs. That is, we use $\pi[m..n]$ to mean $\pi[m]\pi[m+1] \ldots \pi[n-1]$, $\pi[..n]$ for $\pi[0]\pi[1] \ldots \pi[n-1]$ and $\pi[m..]$ for $\pi[m]\pi[m+1] \ldots$. Now, consider a run π. We say that π *visits* a state s if there is some $k \in \mathbb{N}$ such that $\pi[k] = s$. Since there are only finitely many states, some must be visited infinitely often. And, unless all states are visited infinitely often, there will also exist some set of states that are visited only finitely often. Thus, given a run π, we can define the following two sets, which one can use to define objectives over runs: $\mathrm{Inf}(\pi) = \{s \in \mathrm{St} \mid \pi \text{ visits } s \text{ infinitely often}\}$ and its complement $\mathrm{Fin}(\pi) = \mathrm{St} \setminus \mathrm{Inf}(\pi)$.

Strategies. In order to describe how each player plays the game, we need to introduce the concept of a 'strategy'. A strategy for a given player, in its most general form, can be understood as a function, $\sigma_i : \mathrm{St}^+ \to \mathrm{Ac}_i$, which maps sequences, or histories, of states into an action for the player. A strategy profile is a vector of strategies, $\vec{\sigma} = (\sigma_1, \ldots, \sigma_{|\mathrm{Ag}|})$, one for each player. The set of strategies for a given player i is denoted by Σ_i and the set of strategy profiles is denoted by Σ. If we have a strategy profile $\vec{\sigma} = (\sigma_1, \ldots, \sigma_{|\mathrm{Ag}|})$, we use the notation $\vec{\sigma}_{-i}$ to denote the vector $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_{i+1}, \ldots, \sigma_{|\mathrm{Ag}|})$ and $(\vec{\sigma}_{-i}, \sigma_i')$ to denote $(\sigma_1, \ldots, \sigma_{i-1}, \sigma_i', \sigma_{i+1}, \ldots, \sigma_{|\mathrm{Ag}|})$. Finally, we write Σ_{-i} as shorthand for $\Sigma_1 \times \ldots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \ldots \times \Sigma_{|\mathrm{Ag}|}$. All of these notations can also be generalised in the obvious way to *coalitions* of agents, $C \subseteq \mathrm{Ag}$. A strategy profile $\vec{\sigma} \in \Sigma$ together with a state s will induce a unique run, which we denote by $\rho(\vec{\sigma}, s) : \mathbb{N} \to \mathrm{St}$, as well as an infinite sequence of actions $\vec{\mathrm{ac}} : \mathbb{N} \to \mathrm{Ac}$, with $\mathrm{Ac} = \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|}$. These runs are obtained in the following way. Starting from s, each player plays $\mathrm{ac}_i^0 = \sigma_i(s)$. This transforms the game into a new state, given by $s^1 = \mathrm{tr}(s, \mathrm{ac}_1^0, \ldots, \mathrm{ac}_{|\mathrm{Ag}|}^0)$. Each player then plays $\mathrm{ac}_i^1 = \sigma_i(ss^1)$, and this process repeats forever, defining the runs of states and actions. Typically, we are interested in runs that begin in the game's start state, s^0, and we write $\rho(\vec{\sigma})$ as shorthand for the infinite run $\rho(\vec{\sigma}, s^0)$.

It can be useful to work with strategies which are able to be finitely represented. In this work use consider two such representations: *finite-memory strategies* and *memoryless strategies*. A finite-memory strategy is a finite state machine with output: for player i, a finite-memory strategy σ_i is a four-tuple, $(Q_i, q_i^0, \delta_i, \tau_i)$, where, Q_i is a finite, non-empty set of internal states with $q_i^0 \in Q_i$ an initial state, $\delta_i : Q_i \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to Q_i$ is an internal transition function and $\tau_i : Q_i \to \mathrm{Ac}_i$ is an action function. This strategy operates by starting in the initial state, and for each state it is in, producing an action according to τ_i, looking at what actions have been taken by everyone, and then moving to a new internal state as prescribed by δ. Because such a sequence will be periodic, we can write the run induced on the concurrent game structure as $\pi = \pi[..k]\pi[k..m]^\omega$, for some $k, m \in \mathbb{N}$ with $0 \leq k < m$. Finally, a memoryless strategy is a strategy that depends only on the state the player is currently in. Then, it can be written as a function $\sigma_i : \mathrm{St} \to \mathrm{Ac}_i$. Note that memoryless strategies can be encoded as finite-memory strategies, and that finite-memory strategies are a special case of arbitrary strategies $\sigma_i : \mathrm{St}^+ \to \mathrm{Ac}_i$. Whilst we will work with finite-memory and memoryless strategies, we will use arbitrary strategies by default, unless otherwise stated.

Mean-payoff games. A mean-payoff game, G, is given by a tuple, $G = (M, (w_i)_{i \in \mathrm{Ag}})$, where M is a concurrent game structure and for each agent $i \in \mathrm{Ag}$, $w_i : \mathrm{St} \to \mathbb{Z}$ is a weight function [11, 41, 45]. In a mean-payoff game, a run of states, $\pi = s^0 s^1 \ldots$ induces an infinite sequence of weights for each player, $w_i(s^0)w_i(s^1) \ldots$. We denote this sequence by $w_i(\pi)$. Under a given run, π, a player's payoff is given by $\mathrm{mp}(w_i(\pi))$, where for $\beta \in \mathbb{Z}^\omega$, we have $\mathrm{mp}(\beta) = \liminf_{n \to \infty} \frac{1}{n} \sum_{i=0}^{n-1} \beta_i$. For notational convenience, we will write $\mathsf{pay}_i(\pi)$ for $\mathrm{mp}(w_i(\pi))$. We can then define a preference relation over runs for each player as follows: $\pi \succeq_i \pi'$ if and only if $\mathsf{pay}_i(\pi) \geq \mathsf{pay}_i(\pi')$. We also write $\pi \succ_i \pi'$ if $\pi \succeq_i \pi'$ and not $\pi' \succeq_i \pi$.

Solution concepts. We consider solution concepts in the *non-cooperative* and *cooperative* game theory literatures. On one hand, a strategy profile $\vec{\sigma}$ is said to be a *Nash equilibrium* [33, 34] if for all players i and strategies σ_i', we have $\vec{\sigma} \succeq_i (\vec{\sigma}_{-i}, \sigma_i')$. Informally, a Nash equilibrium is a strategy profile from which no player has any incentive to unilaterally deviate. On the other hand, we also consider the *core* [20], a fundamental solution concept that arises from *cooperative* game theory [15]. While Nash equilibria are profiles that are resistant to unilateral deviations, the core consists of profiles that are resistant to those deviations by coalitions of agents, where every member of the coalition is better off, regardless of what the rest of the agents do. Formally, we say that a strategy profile, $\vec{\sigma}$, is in the core if for all coalitions $C \subseteq \mathrm{Ag}$, and strategy vectors $\vec{\sigma}_C'$, then there is some complementary strategy vector $\vec{\sigma}_{\mathrm{Ag} \setminus C}'$ such that $\vec{\sigma} \succeq_i (\vec{\sigma}_C', \vec{\sigma}_{\mathrm{Ag} \setminus C}')$, for some $i \in C$. Given a game G, let $\mathrm{NE}(G)$ denote the set of Nash equilibrium strategy profiles of G, and let $\mathrm{CORE}(G)$ denote the set of strategy profiles in the core of G.

It is worth noting that if a strategy profile is not a Nash equilibrium, then at least one player can deviate and be better off, under the assumption that the remainder of the players do not change their actions. However, if a strategy profile is not in the core, then some coalition can deviate and become better off, *regardless of what* *the other players do*. Thus, the core should not be confused with the solution concept of *strong Nash equilibrium*, which is a strategy profile which is stable under multilateral deviations, assuming the remainder of the players 'stay put' [4, 5]. We do not consider strong Nash equilibria in this work, but simply mention them to further highlight the different game-theoretic nature of the core.

ω-regular specifications. In [6], Boolean combinations of atoms of the form $\mathrm{Inf}(F)$ are used to describe acceptance conditions of arbitrary ω-automata. We use this approach to specify system properties for our games. Formally, the language of ω-regular specifications, α, is defined by the following grammar:

$$\alpha := \mathrm{Inf}(F) \mid \neg\alpha \mid \alpha \wedge \alpha,$$

where F ranges over subsets of St. For notational convenience, we write $\mathrm{Fin}(F)$ as shorthand for $\neg\,\mathrm{Inf}(F)$, $\mathrm{Inf}(\bar{F})$ for $\mathrm{Inf}(\mathrm{St} \setminus F)$ and we define disjunction, $\cdot \vee \cdot$, implication $\cdot \rightarrow \cdot$ and bi-implication $\cdot \leftrightarrow \cdot$ in the usual way. The size of a specification is simply the sum of the sizes of the sets within its atoms. We now talk about what it means for a run to *model* a specification. Let π be a run, F be a subset of St and α, β be arbitrary ω-regular specifications. Then,

- $\pi \models \mathrm{Inf}(F)$, if $\mathrm{Inf}(\pi) \cap F \neq \emptyset$;
- $\pi \models \neg\alpha$, if it is not the case that $\pi \models \varphi$;
- $\pi \models \alpha \wedge \beta$, if $\pi \models \alpha$ and $\pi \models \beta$.

Note that we use Inf in two different, but interrelated senses. First, we use it as an operator over runs, as in $\mathrm{Inf}(\pi)$, to denote the set of states visited infinitely often in a run π, but we also use it as an operator over sets, as in $\mathrm{Inf}(F)$, as an atom in the specifications just defined. The semantics of the latter are defined in terms of the former. We will use these interchangeably: usage will be clear from the context. Using this notation, we can readily define conventional ω-regular winning conditions, as in following table[1]:

Type	Associated Sets	ω-regular specification
Büchi	$F \subseteq \mathrm{St}$	$\mathrm{Inf}(F)$
Gen. Büchi	$(F_k)_{k\in K} \subseteq 2^{\mathrm{St}}$	$\bigwedge_{k\in K} \mathrm{Inf}(F_k)$
Rabin	$(L_i, U_i)_{i\in I} \subseteq 2^{\mathrm{St}} \times 2^{\mathrm{St}}$	$\bigvee_{i\in I} \mathrm{Fin}(L_i) \wedge \mathrm{Inf}(U_i)$
Streett	$(L_j, U_j)_{j\in J} \subseteq 2^{\mathrm{St}} \times 2^{\mathrm{St}}$	$\bigwedge_{j\in J} \mathrm{Fin}(L_j) \vee \mathrm{Inf}(U_j)$
Muller	$(F_k)_{k\in K} \subseteq 2^{\mathrm{St}}$	$\bigvee_{k\in K} \mathrm{Inf}(F_k) \wedge \mathrm{Fin}(\overline{F_k})$

With this defined, we can talk about specifications in the context of games. Let $\vec{\sigma}$ be some strategy profile. Then, $\vec{\sigma}$ induces some run, $\rho(\vec{\sigma})$, and given that ω-regular specifications are defined on runs, we can talk about strategies modelling specifications. But we are not interested in whether arbitrary runs model a given specification – it is more natural in the context of *multi-player games* to ask whether the runs induced by some or all of the equilibria of a game model a specification, both in the non-cooperative and in the cooperative contexts, in particular using solution concepts such as Nash equilibrium and the core, respectively.

Example 2.1. To illustrate the concepts we have laid out so far, we give an example. Suppose we have four delivery robots in a warehouse (given by the coloured triangles in Figure 1), who want to pick up parcels at the pickup points (labelled by the bold **P**s) and drop them off at the delivery points (labelled by the bold **D**s). If a robot is not holding a parcel, and goes to a pickup point, it automatically gets given one. If it has a parcel, and goes to the delivery point, then it loses the parcel, and gains a payoff of 1. And, if two robots collide, by entering the same node at the same time, then they crash, and get a payoff of -999 at every future timestep.

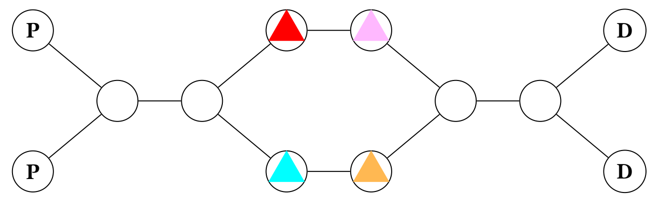

Figure 1: Robots manoeuvering in a warehouse.

Now, there are a number of Nash equilibria here (infinitely many, in fact). But it is easy to see that many of them exhibit undesirable properties. For instance, consider the strategy profile where red and pink go back and forth between the pickup and delivery points, and threaten to crash into, or deadlock, blue and yellow if they move from their starting positions. This is a Nash equilibrium, but is clearly not Pareto optimal – computationally, it is not fair.

It is easy to identify the most socially desirable outcome - all four robots visiting the pickup and delivery points infinitely often, waiting for the others to pass when they reach bottleneck points. If we call the set containing the two states where robots i visits a pickup point P_i and similarly label the set of delivery points D_i, we can express this condition concisely with an ω-regular specification:

$$\bigwedge_{i\in[4]} \mathrm{Inf}(P_i) \wedge \mathrm{Inf}(D_i).$$

Thus, we can conclude that there exists some Nash equilibrium which models the above (generalised Büchi) specification. However, we just did this by inspection. In practice, we would like to ask this question in a more principled way. As such, we will spend the rest of this paper exploring the natural decision problems associated with mean-payoff games with ω-regular specifications.

Before proceeding, we note that given a *fixed* concurrent game structure, not all ω-regular behaviours that the game can exhibit can be described with our formalism (for instance, consider the LTL formula $\boldsymbol{G}\phi$, where ϕ is some propositional formula). However, one can circumvent this restriction by taking the ω-regular automata of the property of interest and performing a product construction with the underlying concurrent game structure.

Mean-payoff games with ω-regular specifications. Given that we have proposed ω-regular specifications as an alternative to LTL [36] specifications, it is natural to ask how they compare. The connection between them is given by the following statement:

Proposition 2.2. *Let G be a game and let α be some ω-regular specification. Then there exists a set of atomic propositions, Φ, a labelling function $\lambda : St \to \mathcal{P}(\Phi)$, and an LTL formula φ such that, for all runs π, we have $\pi \models \alpha$ if and only if $\lambda(\pi) \models \varphi$.*

Thus, on a fixed concurrent game structure, ω-regular specifications can be seen as being 'isomorphic' to a strict subset of LTL. As

[1] We can, of course, also define all other ω-regular properties, including safety, liveness, co-Büchi, and parity acceptance conditions, which due to space are omitted.

such, we hope the restriction of the setting may yield some lower complexities when considering the analogous decision problems. That is, we will study a number of decision problems within the rational verification framework [20, 44], where ω-regular specifications replace LTL specifications in a very natural way.

Firstly, given a game, a strategy profile, and an ω-regular specification, we can ask if the strategy profile is an equilibrium whose induced run models the specification. Secondly, given a game and an ω-regular specification, we can ask if the specification is modelled by the run/runs induced by some/every strategy profile in the set of equilibria of the game. Each of these problems can be phrased in the context of a non-cooperative game or a cooperative game, depending on whether we let the set of equilibria be, respectively, the Nash equilibria or the core of the game. Formally, in the non-cooperative case, we have the following decision problems:

MEMBERSHIP:
Given: Game G, strategy profile $\vec{\sigma}$, and specification, α.
Question: Is it the case that $\vec{\sigma} \in \mathrm{NE}(G)$ and $\rho(\vec{\sigma}) \models \alpha$?

E-NASH:
Given: Game G and specification α.
Question: Does there exist a $\vec{\sigma} \in \mathrm{NE}(G)$ such that $\rho(\vec{\sigma}) \models \alpha$?

A natural dual to **E-NASH** is the **A-NASH** problem, which instead of asking if the specification holds in the run induced by *some* Nash equilibrium, asks if the specification holds in *all* equilibria. Formally, this decision problem is stated as follows:

A-NASH:
Given: Game G and specification α.
Question: Is it the case that $\rho(\vec{\sigma}) \models \alpha$, for all $\vec{\sigma} \in \mathrm{NE}(G)$?

In the cooperative setting, the analogous problems are defined simply by changing $\mathrm{NE}(G)$ for $\mathrm{CORE}(G)$. They are called **MEMBERSHIP**, **E-CORE**, and **A-CORE**, respectively – as can be seen, with a small abuse of notation for the first decision problem.

It is worth noting here one technical detail about representations. In the **E-NASH** problem, the quantifier asks if there exists a Nash equilibrium which models the specification. This quantification ranges over all possible Nash equilibria and the strategies may be arbitrary strategies. However, in the **MEMBERSHIP** problem, the strategy $\vec{\sigma}$ is part of the input, and thus, needs to be finitely representable. Therefore, when considering **E-NASH** (or **A-NASH**, or the corresponding problems for the core), we place no restrictions on the strategies, but when reasoning about **MEMBERSHIP**, we work exclusively with memoryless or finite-memory strategies.

3 NON-COOPERATIVE GAMES

In the non-cooperative setting, **MEMBERSHIP**, **E-NASH**, and **A-NASH** are the relevant decision problems. In this section, we will show that **MEMBERSHIP** lies in P for memoryless strategies, while **E-NASH** is NP-complete for memoryless, finite-memory strategies, as well as arbitrary strategies – thus, no worse than solving a multi-player mean-payoff game [41]. Because **A-NASH** is the dual problem of **E-NASH**, it also follows that **A-NASH** is co-NP-complete. In order to obtain some of these results, we also provide a semantic characterisation of the runs associated with strategy profiles in the set of Nash equilibria that satisfy a given ω-regular specification. We will first study the **MEMBERSHIP** problem, and then investigate **E-NASH**, providing an upper bound for arbitrary strategies and a lower bound for memoryless strategies.

Proposition 3.1. *For memoryless strategies, **MEMBERSHIP** is in* P.

Proof Sketch. To demonstrate it is a Nash equilibrium, we 'run' the strategy profile to calculate each player's payoff and in the process, we can verify it models the specification. Then for each player, we fix the strategy vector of the other players, and then use Karp's algorithm [31] to verify that the given player has no other memoryless strategy under which they are better off. □

Given the simplicity of the above algorithm, there is some hope that is might extend to finite-memory strategies. However, in this case, the entire configuration of the game is not just given by the current state – it is given by the current state, as well as the state that each of the player's strategies are in. Thus, we might have to visit at least $|\mathrm{St}| \cdot |Q|^{|\mathrm{Ag}|} + 1$ (where Q is the smallest set of strategy states over the set of players) configurations until we discover a loop. This quantity is not polynomial in the size of the input, and so we cannot use the above algorithm in the case of finite memory strategies to get a polynomial time upper bound.

We now consider **E-NASH**. Instead of providing the full NP-completeness result here, we start by showing that the problem is NP-hard, even for memoryless strategies, and delay the proof of the upper bound until we develop a useful characterisation of Nash equilibrium in the context of ω-regular specifications. For now, we have the following hardness result, obtained using a reduction from the Hamiltonian cycle problem [14, 30] – a similar, but simpler, reasoning technique can be found in [41].

Proposition 3.2. ***E-NASH** is* NP-hard*, even for games with one player, constant weights, and memoryless strategies.*

Propositions 3.1 and 3.2 together give an NP-completeness result for multi-player mean-payoff games with ω-regular specifications and memoryless strategies: one can non-deterministically guess a memoryless strategy for each player (which is simply a list of actions for each player, one for each state), and use **MEMBERSHIP** to verify that it is indeed a Nash equilibrium that models the specification. However, as shown later, the problem is NP-complete in the general case, which we show using the characterisation below.

To characterise the Nash equilibria of these games we need to introduce the notion of the *punishment value* in a multi-player mean-payoff game [21, 22]. The punishment value, $\mathrm{pun}_i(s)$, of a player i in a given state s can be thought of as the worst value the other players can impose on a player at a given state. Concretely, if we regard the game G as a two player, zero-sum game, where player i plays against the coalition $\mathrm{Ag} \setminus \{i\}$, then the punishment value for player i is the smallest mean-payoff value that the rest of players in Ag can inflict on i from a given state. Formally, given a player i and a state $s \in \mathrm{St}$, we define the punishment value, $\mathrm{pun}_i(s)$ against player i at state s, as follows:

$$\mathrm{pun}_i(s) = \min_{\vec{\sigma}_{-i} \in \Sigma_{-i}} \max_{\tau_i \in \Sigma_i} \mathrm{pay}_i(\rho((\vec{\sigma}_{-i}, \tau_i), s))$$

How efficiently can we calculate this value? As established in [41], we proceed in the following way: in a two player, turn-based, zero-sum, mean-payoff game, positional strategies suffice to achieve the punishment value [11]. Thus, we can non-deterministically guess a pair of positional strategies for each player (one for the coalition punishing the player, and one for the player themselves), use Karp's algorithm [31] to find the maximum payoff for both the player and the coalition against their respective punishing strategies, and then verify that the two values coincide. With this established, we have the following lemma, which can be proved using techniques for mean-payoff games adapted from [25, 41].

Lemma 3.3. *Let π be a run in G and let $\{\vec{ac}[k]\}_{k\in\mathbb{N}}$ be the run of associated action profiles. Then there is a Nash equilibrium, $\vec{\sigma} \in NE(G)$, such that $\pi = \rho(\vec{\sigma})$ if and only if there exists some $\vec{z} \in \mathbb{Q}^{Ag}$, with $z_i \in \{pun_i(s) \mid s \in St\}$, such that:*

- *for each k, we have $pun_i(tr(\pi[k], (\vec{ac}[k]_{-i}, ac_i'))) \leq z_i$ for all $i \in Ag$ and $ac_i' \in Ac_i$, and;*
- *for all players $i \in Ag$, we have $z_i \leq pay_i(\pi)$.*

With this lemma in mind, we define a graph, $G[\vec{z}; F] = (V, E)$ as follows. We set $V = \mathrm{St}$ and include $e = (u, v) \in E$ if there exists some action profile $\vec{\mathrm{ac}}$ such that $v = \mathrm{tr}(u, \vec{\mathrm{ac}})$ with $\mathrm{pun}_i(\mathrm{tr}(u, (\vec{\mathrm{ac}}_{-i}, \mathrm{ac}_i'))) \leq z_i$ for all $i \in \mathrm{Ag}$ and $\mathrm{ac}_i' \in \mathrm{Ac}_i$. Having done this, we then prune any components which cannot be reached from the start state and then remove all states and edges not contained in F, before reintroducing any states in F that may have been removed. Thus, given this definition and the preceding lemma, to determine if there exists a Nash equilibrium which satisfies an ω-regular specification, α, we calculate the punishment values, and guess a vector $\vec{z_s} \in \mathrm{St}^{\mathrm{Ag}}$, as well a set of states, F, which satisfy the specification. Letting $z_i = \mathrm{pun}_i(z_s)$, we form the graph $G[\vec{z}; F]$ and then check if there is some run π in $G[\vec{z}; F]$ with $z_i \leq \mathrm{pay}_i(\pi)$ for each player i which visits every state infinitely often. Trivially, if this graph is not strongly connected, then no run can visit every state infinitely often. Thus, to determine if the above condition holds, we need one more piece of technical machinery, in the form of the following proposition:

Proposition 3.4. *Let $G = (V, E)$ be a strongly connected graph, let $\{w_i\}_{i\in Ag}$ be a set of weight functions, let $\vec{z} \in \mathbb{Q}^{Ag}$. Then, we can determine if there is some run π such that i) $z_i \leq pay_i(\pi)$ for each $i \in Ag$ and ii) visits every state infinitely often, in polynomial time.*

Conceptually, Proposition 3.4 is similar to Theorem 18 of [41], but with two keys differences - firstly, we need to do additional work to determine if there is a path that visits every state infinitely often. Moreover, the argument of [41] is adapted so we have the corollary that if there is a Nash equilibrium that models the specification, then there is some finite state Nash equilibrium that also models the specification. This means that the construction in our proof can not only be used for verification, but also for synthesis.

With the above series of propositions in place, we are now ready to establish the complexity of the **E-NASH** problem.

Proposition 3.5. ***E-NASH** is* NP-complete.

Proof. For NP-hardness we have Proposition 3.2. For the upper bound, suppose we have an instance, (G, α), of the problem. Then we proceed as follows. We non-deterministically guess pairs of punishing strategy profiles, $(\zeta_i, \vec{\zeta}_{-i})$ for each player $i \in \mathrm{Ag}$, a state z_s for each player, and a set of states F. From these, we can easily check that the valuation induced by F satisfies the specification and we can also use Karp's algorithm to compute the punishment values, $\mathrm{pun}_i(s)$, for each state $s \in \mathrm{St}$ and for each player $i \in \mathrm{Ag}$. Setting $z_i = \mathrm{pun}_i(z_s)$, we invoke Lemma 3.3 and form the graph $G[\vec{z}; F]$. If it is not strongly connected, then we reject. Otherwise, we use Proposition 3.4 to determine if the associated linear program has a solution. If it does, then we accept, otherwise we reject. □

Corollary 3.6. *Let G be a game and α an ω-regular specification. Suppose that G has some Nash equilibrium $\vec{\sigma}$ such that $\vec{\sigma} \models \alpha$. Then, G also has some finite-memory Nash equilibrium $\vec{\sigma}'$ such that $\vec{\sigma}' \models \alpha$.*

4 COOPERATIVE GAMES

We begin by asking whether games always have a non-empty core, a property that holds for games with LTL goals and specifications [20]. We find that this does not hold in general for mean-payoff games.

Proposition 4.1. *In mean-payoff games, if $|Ag| \leq 2$, then the core is non-empty. For $|Ag| > 2$, there exist games with an empty core.*

The proof of the above for the two-player case is routine manipulation. The counterexample for when $|\mathrm{Ag}| > 2$ is omitted due to space, but consists of a start state leading to three sink states, with three players. The game is such that there are always two players who can beneficially deviate from any given state.

Before proceeding, it is worth reflecting on the definition of the core. We can redefine this solution concept in the language of 'beneficial deviations'. That is, we say that given a game G, a strategy profile $\vec{\sigma}$, a *beneficial deviation* by a coalition C, is a strategy vector $\vec{\sigma}_C'$ such that for all complementary strategy profiles $\vec{\sigma}_{\mathrm{Ag}\setminus C}'$, we have $\rho(\vec{\sigma}_C', \vec{\sigma}_{\mathrm{Ag}\setminus C}') \succ_i \rho(\vec{\sigma})$ for all $i \in C$. We can then say that $\vec{\sigma}$ is a member of the core, if there exists no coalition C which has a beneficial deviation from $\vec{\sigma}$. Note this formulation is entirely identical to our earlier definition of the core.

From a computational perspective, there is an immediate concern here - given a potential beneficial deviation, how can we verify that it is preferable to the status quo under *all* possible counter-responses? Given that strategies can be arbitrary mathematical functions, how can we reason about that universal quantification effectively? Fortunately, as we show in the following lemma, we can restrict our attention to memoryless strategies when thinking about potential counter-responses to players' deviations:

Lemma 4.2. *Let G be a game, $C \subseteq Ag$ be a coalition and $\vec{\sigma}$ be a strategy profile. Further suppose that $\vec{\sigma}_C'$ is a strategy vector such that for all memoryless strategy vectors $\vec{\sigma}_{Ag\setminus C}'$, we have,*

$$\rho(\vec{\sigma}_C', \vec{\sigma}_{Ag\setminus C}') \succ_i \rho(\vec{\sigma}).$$

Then, for all strategy vectors, $\vec{\sigma}_{Ag\setminus C}'$, not necessarily memoryless, we have,

$$\rho(\vec{\sigma}_C', \vec{\sigma}_{Ag\setminus C}') \succ_i \rho(\vec{\sigma}).$$

Before we prove this, we need to introduce an auxillary concept of *two-player, turn-based, zero-sum, multi-mean-payoff games* [43] (we will just call these multi-mean-payoff games moving forward). Informally, these are similar to two-player, turn-based, zero-sum mean-payoff games, except player 1 has k weight functions associated with the edges, and they are trying to ensure the resulting

k-vector of mean-payoffs is component-wise greater than a vector threshold. Formally, a multi-mean-payoff game is a 5-tuple, $G = (V_1, V_2, v^0, E, w, z^k)$, where V_1, V_2 are sets of states controlled by players 1 and 2 respectively, with $V = V_1 \cup V_2$ the state space, $v^0 \in V$ the start state, $E \subseteq V \times V$ a set of edges and $w : E \to \mathbb{Z}^k$ a weight function, assigning to each edge a vector of weights.

The game is played by starting in the start state, $s^0 \in S_i$, and player i choosing an edge (s^0, s^1), and traversing it to the next state. From this new state, $s^1 \in S_j$, player j chooses an edge and so on, repeating this process forever. Runs are defined in the usual way and the payoff of a run π, $\mathsf{pay}(\pi)$, is simply the vector $(\mathsf{mp}(w_1(\pi)), \ldots, \mathsf{mp}(w_k(\pi)))$. Finally, $z^k \in \mathbb{Q}^k$ is a threshold vector and player 1 wins if the $\mathsf{pay}_i(\pi) \geq z_i$ for all $i \in \{1, \ldots, k\}$, and loses otherwise. An important question associated with these games is whether player 1 can force a win. As shown in [43], this problem is co-NP-complete. Whilst we do not need to utilise this result right now, this sets us up to prove Lemma 4.2:

Proof of Lemma 4.2. Let $\vec{\sigma}_{\mathrm{Ag}\setminus C}$ be an arbitrary strategy and let $i \in C$ be an arbitrary agent. Denote $\rho(\vec{\sigma})$ by ρ and $\rho(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$ by ρ'. We aim to show that $\rho' \succ_i \rho$. Suppose instead it is the case that $\rho \succeq_i \rho'$. Thus, we have $\rho(\vec{\sigma}) \succeq_i \rho(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$. Considering this as a two-player multi-mean-payoff game, where player 1's strategy is fixed and encoded into the game structure (*i.e.*, player 1 follows $\vec{\sigma}'_C$, but has no say in the matter), and the payoff threshold is $\mathsf{mp}(\rho(\vec{\sigma}))$, then $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$ is a winning strategy for player 2 in this game. Now, by [32, 43], if player 2 has a winning strategy, then they have a memoryless winning strategy. Thus, there is a memoryless strategy $\vec{\sigma}''_{\mathrm{Ag}\setminus C}$ such that $\rho(\vec{\sigma}) \succeq_i \rho(\vec{\sigma}'_C, \vec{\sigma}''_{\mathrm{Ag}\setminus C})$. But this contradicts the assumptions of the lemma, and thus we must have $\rho' \succ_i \rho$. □

We now look at some complexity bounds for mean-payoff games in the cooperative setting. Having introduced beneficial deviations, let us consider the following decision problem:

BENEFICIAL-DEVIATION (BEN-DEV):
Given: Game G and strategy profile $\vec{\sigma}$.
Question: Is there $C \subseteq \mathrm{Ag}$ and $\vec{\sigma}'_C \in \Sigma_C$ such that for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C} \in \Sigma_{\mathrm{Ag}\setminus C}$ and for all $i \in C$, we have:

$$\rho(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}) \succ_i \rho(\vec{\sigma})?$$

Using this new problem, we can prove the following statement.

Proposition 4.3. *Let G be a game, $\vec{\sigma}$ a strategy profile, and α a specification. Then,* $(G, \vec{\sigma}, \alpha) \in$ **MEMBERSHIP** *if and only if* $(G, \vec{\sigma}) \notin$ **BEN-DEV** *and* $\rho(\vec{\sigma}) \models \alpha$.

The above proposition characterises the **MEMBERSHIP** problem for cooperative games in terms of beneficial deviations, and, in turn, provides a direct way to study its complexity. In the remainder of this section we concentrate on the memoryless case.

Proposition 4.4. *For memoryless strategies,* **BEN-DEV** *is* NP-complete.

Proof. First correctly guess a deviating coalition C and a strategy profile $\vec{\sigma}'_C$ for such a coalition of players. Then, use the following three-step algorithm. First, compute the mean-payoffs that players in C get on $\rho(\vec{\sigma})$, that is, a set of values $z^*_j = \mathsf{pay}_j(\rho(\vec{\sigma}))$ for every $j \in C$ — this can be done in polynomial time simply by 'running' the strategy profile $\vec{\sigma}$. Then compute the graph $G[\vec{\sigma}'_C]$, which contains all possible behaviours (*i.e.*, strategy profiles) for $\mathrm{Ag} \setminus C$ with respect to $\vec{\sigma}$ - this construction is similar to the one used in the proof of Proposition 3.1, that is, the game when we fix $\vec{\sigma}'_C$, and can be done in polynomial time. Finally, we ask whether every path π in $G[\vec{\sigma}'_C]$ satisfies $\mathsf{pay}_j(\pi) > z^*_j$, for every $j \in C$ - for this step, we can use Karp's algorithm to answer the question in polynomial time for every $j \in C$. If every path in $G[\vec{\sigma}'_C]$ has this property, then we accept; otherwise, we reject. For hardness, we reduce from 3SAT, using a variation of the construction in [38]. □

From Proposition 4.4 follows that checking if no coalition of players has a beneficial deviation with respect to a given strategy profile is a co-NP problem. More importantly, it also follows that **MEMBERSHIP** is a co-NP-complete problem too.

Proposition 4.5. *For memoryless strategies,* **MEMBERSHIP** *is* co-NP-complete.

Proof. Recall that given a game G, a strategy profile $\vec{\sigma}$, and an ω-regular specification α, we have $(G, \vec{\sigma}, \alpha) \in$ **MEMBERSHIP** if and only if $(G, \vec{\sigma}) \notin$ **BEN-DEV** and $\rho(\vec{\sigma}) \models \alpha$. Thus, we can solve **MEMBERSHIP** simply by first checking $\rho(\vec{\sigma}) \models \alpha$, which can be done in polynomial time and we reject if that check fails. If $\rho(\vec{\sigma}) \models \alpha$, then we ask $(G, \vec{\sigma}) \in$ **BEN-DEV** and accept if that check fails, and reject otherwise. Finally, since **BEN-DEV** is NP-hard, it follows from the above procedure that **MEMBERSHIP** is co-NP-hard, which concludes the proof of the statement. □

BEN-DEV can also be used to solve **E-CORE** in this case.

Proposition 4.6. *For memoryless strategies,* **E-CORE** *is in* Σ^{P}_2.

Proof. Given any instance (G, α), we guess a strategy profile $\vec{\sigma}$ and check that $\rho(\vec{\sigma}) \models \alpha$ and that $(G, \vec{\sigma}, \alpha)$ is not an instance of **BEN-DEV**. While the former can be done in polynomial time, the latter can be solved in co-NP using an oracle for **BEN-DEV**. Thus, we have a procedure that runs in $\mathrm{NP}^{\text{co-NP}} = \mathrm{NP}^{\mathrm{NP}} = \Sigma^{\mathrm{P}}_2$. □

Proposition 4.6 sharply contrasts with that for Nash equilibrium, where the same problem lies in NP. More importantly, the result also shows that the (complexity) dependence on the type of coalitional deviation is only weak, in the sense that different types of beneficial deviations may be considered within the same complexity class, as long as such deviations can be checked with an NP or co-NP oracle. For instance, in [20] other types of cooperative solution concepts are defined, which differ from the one in this paper (known in the cooperative game theory literature as α-core [35]) simply in the type of beneficial deviation under consideration. Another concept introduced in [20] is that of 'fulfilled coalition', which informally characterises coalitions that have the strategic power (a joint strategy) to ensure a minimum given payoff *no matter what the other players in the game do*. Generalising to our setting, from qualitative to quantitative payoffs, we introduce the notion of a *lower bound*, which we will use to reason about cooperative games.

Definition 4.7. Let $C \subseteq \mathrm{Ag}$ be a coalition in a game G and let $\vec{z}_C \in \mathbb{Q}^C$. We say that $\vec{z}_C = (z_1, \ldots, z_i, \ldots, z_{|C|})$ is a *lower bound* for C if there is a joint strategy $\vec{\sigma}_C$ for C such that for all strategies $\vec{\sigma}_{-C}$ for $\mathrm{Ag} \setminus C$, we have $\mathsf{pay}_i(\rho(\vec{\sigma}_C, \vec{\sigma}_{-C})) \geq z_i$, for every $i \in C$.

Based on the definition above, we can prove the following lemma, which characterises the core in terms of runs where mean-payoffs can be ensured collectively, no matter any adversarial behaviour.

Lemma 4.8. *Let π be a run in G. There is $\vec{\sigma} \in CORE(G)$ such that $\pi = \rho(\vec{\sigma})$ if and only if for every coalition $C \subseteq Ag$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, there is some $i \in C$ such that $z_i \leq pay_i(\pi)$.*

With this lemma in mind, we want to determine if a given vector, $\vec{z}_C$, is in fact a lower bound and importantly, how efficiently we can do this. That is, to understand the following decision problem:

LOWER-BOUND:
Given: Game G, coalition $C \subseteq \text{Ag}$, and vector $\vec{z}_C \in \mathbb{Q}^{\text{Ag}}$.
Question: Is $\vec{z}_C$ is a lower bound for C in G?

Proposition 4.9. ***LOWER-BOUND** is co-NP-complete.*

Whilst the results thus far give us key insights into the nature of the core, a general upper bound for **E-CORE** remains elusive.

5 WEIGHTED REACTIVE MODULE GAMES

One problem with concurrent game structures as we have worked with them so far is that they are extremely verbose. The transition function, $\text{tr} : \text{St} \times \text{Ac}_1 \times \cdots \times \text{Ac}_{|\text{Ag}|} \to \text{St}$ is a total function, so it has size $|\text{Ac}|^{|\text{Ag}|}$. Thus, the size of the game scales exponentially with the number of the agents. In example 2.1, the underlying concurrent game structure has a size of 429,981,696. Obviously, such a simple example can (and should) be specified in a much more concise way.

One natural framework we can use to induce succinctness is that of *Reactive Modules* [2]. Specifically, we modify the Reactive Module Games of [18] with weights on the guarded commands. We begin by walking through some preliminaries.

Reactive Module Games do not use the full power of reactive modules, but instead use a subset of the reactive modules syntax, namely the *simple reactive modules language* (SRML) [42]. In SRML terms, agents are described by *modules*, which in turn consist of a set of variables controlled by the module, along with a set of *guarded commands*. Formally, given a set of propositional variables Φ, a guarded command g is an expression of the form,

$$\varphi \rightsquigarrow x'_1 := \psi_1; \ldots; x'_k := \psi_k,$$

where φ and each ψ_i are propositional formulae over Φ and each x_j also lies in Φ. We call φ the *guard* of g and denote it by $\text{guard}(g)$, and we call the variables (the x_js) on the right-hand-side of g the *controlled variables* of g, denoted by $\text{ctr}(g)$. The idea is that under a given valuation of a set of variables, $v \subseteq \Phi$, each module has a set of commands for which $\text{guard}(g)$ is true (we say that they are enabled for execution). Each module can then choose one enabled command, g, and reassign the variables in $\text{ctr}(g)$ according to the assignments given on the right hand side of g. For instance, if φ were true, then the above guarded command could be executed, setting each x_j to the truth value of ψ_j under v. Only if no g is enabled, a special guarded command $g^{\mathbf{skip}}$ – which does not change the value of any controlled variable – is enabled for execution so that modules always have an action they can take.

Given a set of propositional variables, Φ, a simple reactive module, m, is a tuple (Ψ, I, U), where,

- $\Psi \subseteq \Phi$ is a set of propositional variables;
- I is a set of *initialisation* guarded commands, where for all $g \in I$, we have $\text{guard}(g) = \top$ and $\text{ctr}(g) \subseteq \Psi$.
- U is a set of *update* guarded commands, where for all $g \in U$, $\text{guard}(g)$ is a propositional formula over Φ and $\text{ctr}(g) \subseteq \Psi$.

An SRML arena, A, is a tuple $A = (\text{Ag}, \Phi, \{m_i\}_{i \in \text{Ag}})$, where Ag is a finite, non-empty set of agents, Φ is a set of propositional variables and each m_i is a simple reactive module $m_i = (\Phi_i, I_i, U_i)$ such that $\{\Phi_i\}_{i \in \text{Ag}}$ is a partition for Φ. With this syntactic machinery in place, we are finally ready to describe the semantics of SRML arenas. In the interest of conciseness, we give a brief, high-level description here – for full mathematical details, please refer to [18, 42].

Given a valuation $v \subseteq \Phi$ at a point in time, each agent i has a set of commands they can use, denoted $\text{enabled}_i(v)$. We then denote the set of possible vectors of guarded commands across all players under a given valuation by $\text{enabled}(v)$. Given a valuation v, and a joint guarded command $J \in \text{enabled}(v)$, the new valuation induced by executing this command is denoted $\text{exec}(J, v)$.

The game starts by each agent choosing an initialisation command, which induces a first valuation v^0. Each player then picks a guarded command in $\text{enabled}_i(v^0)$, forming a joint guarded command, J^1, and the game moves to the valuation $v^1 = \text{exec}(J^1, v^0)$. This process repeats ad infinitum, producing a run of the game as before. However, unlike in the previous setting, where we defined runs over states of the game, here, we define runs over joint guarded commands, $\rho : \mathbb{N} \to (I_1 \cup U_1) \times \ldots \times (I_{|\text{Ag}|} \cup U_{|\text{Ag}|})$. Whilst, superficially, this may look like a departure from our previous convention, it is not. Given that these games are entirely deterministic, if we know the sequence of joint guarded commands that have been taken, we can infer the sequence of states. Additionally, knowing the sequence of joint guarded commands provides us with more information that knowing the sequence of states - a state may have multiple joint guarded commands that lead to it. All of the techniques we developed before transfer readily to this new setting, so take it for granted that there is a straightforward link between the two approaches and will not comment on it further.

We can now define *weighted reactive module games.* A weighted reactive module game (WRMG), $G = (A, \{w_i\}_{i \in \text{Ag}})$, is an SRML arena, $A = (\text{Ag}, \Phi, \{m_i\}_{i \in \text{Ag}})$, along with a set of weight functions, with $w_i : I_i \cup U_i \to \mathbb{Z}$. That is, each module has an assigned weight function that maps commands to integers. As before, a player's payoff is given by the mean-payoff of the weights attached to a run.

Finally, we need to define ω-regular specifications in the context of WRMGs. Sets of states are already conveniently parameterised by the propositional variables of Φ, so we introduce specifications which are Boolean combinations of atoms of the form $\text{Inf}(p)$ with $p \in \Phi$. The semantics of these specifications are defined in a nearly identical way to before. Let us now walk through an example to demonstrate their conciseness and utility of WRMGs.

Example 5.1. In the robot example from before (Example 2.1), the state of the game is entirely described by the position of each of the four robots, whether they are holding a parcel or not, and whether they have crashed. Thus, we define four reactive modules $m_1, \ldots, m_4$ with $m_i = (\Phi_i, I_i, U_i)$ as follows - we set $\Phi_i = \{x_{i,1}, \ldots, x_{i,12}, p_i, c_i\}$, where the x_i model which node the robot is in, numbered top-to-bottom, left-to-right with respect to the diagram, p_i denotes if the robot is carrying a parcel or not, and c_i

denotes if the robot has crashed or not. With this defined, we can define one initialisation command for each robot:

$$\top \rightsquigarrow x'_{i,1} := \bot; \dots x'_{i,n} := \top; \dots; x'_{i,12} := \bot; p'_i := \bot; c'_i := \bot \quad [0],$$

where n is appropriately set, given the starting position of the robot. Additionally, the [0] at the end of the guarded command denotes the weight rewarded for performing that command. Then for each agent i and edge (x_n, x_m) of the graph, we define a guarded command,

$$\neg c_i \wedge x_{i,n} \rightsquigarrow x'_{i,n} := \bot; x'_{i,m} := \top \quad [0].$$

We also model picking up and delivering a parcel, as well as crashing into another robot. We do this with the following commands:

$$\begin{aligned} \neg c_i \wedge \neg p_i \wedge (x_{i,1} \vee x_{i,2}) &\rightsquigarrow p_i\prime := \top \quad [0], \\ \neg c_i \wedge p_i \wedge (x_{i,11} \vee x_{i,12}) &\rightsquigarrow p_i\prime := \bot \quad [1], \\ \neg c_i \wedge x_{i,n} \wedge (x_{j,n} \vee x_{k,n} \vee x_{l_n}) &\rightsquigarrow c'_i := \top \quad [-999] \\ c_i &\rightsquigarrow c'_i := \top \quad [-999] \end{aligned}$$

where i ranges over players; j, k and l range over the other players; and j ranges from 1 to 12. We also have the $g^{\mathbf{skip}}$ command from before, so the robot can stay still on a node for a time step.

It is easy to see that this setup models the example from before, is *exponentially more concise*, requiring 52 guarded commands in total, and is natural to work with. Note that we could save even more space by encoding the robots positions in binary, at the expense of making our guarded commands slightly more complicated. Whilst this technique may be useful for larger systems, we give a unary encoding here for clarity.

With WRMGs now adequately motivated, the main decision problem to consider then is the following:

WRMG-E-NASH:
Given: WRMG G, and ω-regular specification α.
Question: Does there exist a $\vec{\sigma} \in \mathrm{NE}(G)$ such that $\rho(\vec{\sigma}) \models \alpha$?

This problem seems to be harder than answering the same question for games with an explicit representation, *e.g.*, using concurrent games structures. In fact, we have the following result.

Proposition 5.2. *The **WRMG-E-NASH** problem lies in NEXPTIME and is EXPTIME-hard.*

Proof Sketch. For the upper bound, the idea is to 'blow up' the simple reactive module arena into a concurrent game structure, then apply the same techniques as in Section 3. For the lower bound, we reduce from PEEK-G4, known to be EXPTIME-hard [39] □

6 RELATED WORK

Mean-payoff games. Mean-payoff games are a useful verification tool in the analysis of quantitative aspects of computer systems. Most work has been devoted to the study of two-player zero-sum games, which can be solved in NP∩co-NP [45]. Beyond such games, two kinds of mean-payoff games have been studied: multi-player mean-payoff games, whose solution was studied with respect to Nash equilibria [41], and two-player multi-mean-payoff games [43], where the focus is on the computation of winning strategies for either player, a problem that can be solved in NP for memoryless strategies and in co-NP for arbitrary strategies, although in such a case optimal strategies may require infinite memory [43].

Combined qualitative and quantitative reasoning. Combining qualitative and quantitative reasoning has mainly been done by modifying players' mean-payoff with some qualitiatve measure. In [10], the authors consider two-player, zero-sum games, where on each run of the game, every player is assigned a two-size tuple (parity goal, mean-payoff), where each player's payoff is $-\infty$ if the parity goal is not met, and the mean-payoff otherwise. In a similar setting, [21, 22] look at multi-player concurrent games with lexicographic preferences over (parity/LTL goal, mean-payoff) tuples and look at the decision problem of determining if there exists some finite state strict ϵ Nash equilibrium. Additionally, [25] considered multi-player concurrent games where the players have mean-payoff goals, and the question is whether there is some Nash equilibrium which models some temporal specification.

On ω-regular specifications. Games with ω-regular objectives have been studied mostly in the context of two-player games [9], where the goal of one of the players is to show that the ω-regular objective holds in the system, while the goal of the other player is to show otherwise. Such games are usually used in the context of synthesis and model-checking of temporal logic specifications. These two-player zero-sum games are rather different from ours since in our games, the ω-regular specification is not part of the goal of the players, but rather a property that an *external* system designer wishes to see satisfied. This changes completely the overall problem setup and explains why the drastic differences in complexity between traditional games with ω-regular objectives – whose complexity can range from P (for instance, for Büchi games) to PSPACE (for instance, for Muller games) – and multi-player mean-payoff games with ω-regular specifications, even for two-player zero-sum instances with constant weights.

On Rational verification. The problem we have studied in this paper is called Rational Verification, which has been studied for different types of arena games [8, 17, 18, 23], strategies [16], and specification languages, including LTL [17, 18, 23], CTL [19], and LDL [28]. While rational verification is 2EXPTIME-complete with LTL goals, and even undecidable for games with imperfect information [27], the problem can be shown to be considerably easier when considering simpler specification languages [25]. However, in the context of multi-player mean-payoff games, only a solution for generalised Büchi goals was known, using an encoding via GR(1) specifications, and only for Nash equilibrium. In this paper, we have extended such results to account for all ω-regular specifications, and have provided results for cooperative games and succinct representations. With respect to the former, the only relevant related work is [20], where the core for concurrent game structures was introduced. And, regarding the latter, a comprehensive study using reactive modules games can be found in [18] – work that has been extended, and parts of it implemented, in various ways [24, 40].

ACKNOWLEDGMENTS

T. Steeples gratefully acknowledges the support of the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems EP/L015897/1 and the Ian Palmer Memorial Scholarship. M. Wooldridge was supported by JP Morgan and the Alan Turing Institute. We would also like to thank the three anonymous reviewers for their detailed comments and constructive criticism – these were incredibly helpful in shaping the final version of this paper.

REFERENCES

[1] Luca de Alfaro and Thomas A. Henzinger. 2000. Concurrent Omega-Regular Games. In *15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, California, USA, June 26-29, 2000*. IEEE Computer Society, 141–154. https://doi.org/10.1109/LICS.2000.855763

[2] Rajeev Alur and Thomas A. Henzinger. 1996. Reactive Modules. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*. IEEE Computer Society, 207–218. https://doi.org/10.1109/LICS.1996.561320

[3] Rajeev Alur, Thomas A. Henzinger, and Orna Kupferman. 2002. Alternating-time temporal logic. *J. ACM* 49, 5 (2002), 672–713. https://doi.org/10.1145/585265.585270

[4] Robert J. Aumann. 1959. Acceptable points in general cooperative n-person games. In *Contributions to the theory of games, Vol. IV*. Princeton University Press, Princeton, N.J., 287–324.

[5] Robert J. Aumann. 1960. Acceptable points in games of perfect information. *Pacific J. Math.* 10 (1960), 381–417.

[6] Tomás Babiak, Frantisek Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Kretínský, David Müller, David Parker, and Jan Strejcek. 2015. The Hanoi Omega-Automata Format. In *Computer Aided Verification - 27th International Conference, CAV 2015, San Francisco, CA, USA, July 18-24, 2015, Proceedings, Part I (Lecture Notes in Computer Science, Vol. 9206)*, Daniel Kroening and Corina S. Pasareanu (Eds.). Springer, 479–486. https://doi.org/10.1007/978-3-319-21690-4_31

[7] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli, and Yaniv Sa'ar. 2012. Synthesis of Reactive(1) designs. *J. Comput. Syst. Sci.* 78, 3 (2012), 911–938. https://doi.org/10.1016/j.jcss.2011.08.007

[8] Julian C. Bradfield, Julian Gutierrez, and Michael J. Wooldridge. 2016. Partial-order Boolean games: informational independence in a logic-based model of strategic interaction. *Synth.* 193, 3 (2016), 781–811.

[9] Krishnendu Chatterjee and Thomas A. Henzinger. 2012. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.* 78, 2 (2012), 394–413. https://doi.org/10.1016/j.jcss.2011.05.002

[10] Krishnendu Chatterjee, Thomas A Henzinger, and Marcin Jurdzinski. 2005. Mean-payoff parity games. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS'05)*. IEEE, 178–187.

[11] Andrzej Ehrenfeucht and Jan Mycielski. 1979. Positional strategies for mean payoff games. *International Journal of Game Theory* 8, 2 (1979), 109–113.

[12] Dana Fisman, Orna Kupferman, and Yoad Lustig. 2009. Rational Synthesis. *CoRR* abs/0907.3019 (2009). http://arxiv.org/abs/0907.3019 _eprint: 0907.3019.

[13] Tong Gao, Julian Gutierrez, and Michael J. Wooldridge. 2017. Iterated Boolean Games for Rational Verification. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2017, São Paulo, Brazil, May 8-12, 2017*. ACM, 705–713.

[14] M. R. Garey and David S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman.

[15] Donald B. Gillies. 1959. 3. Solutions to General Non-Zero-Sum Games. In *Contributions to the Theory of Games (AM-40), Volume IV*, Albert William Tucker and Robert Duncan Luce (Eds.). Princeton University Press, 47–86. https://doi.org/10.1515/9781400882168-005

[16] Julian Gutierrez, Paul Harrenstein, Giuseppe Perelli, and Michael J. Wooldridge. 2019. Nash Equilibrium and Bisimulation Invariance. *Log. Methods Comput. Sci.* 15, 3 (2019).

[17] Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. 2015. Iterated Boolean games. *Inf. Comput.* 242 (2015), 53–79. https://doi.org/10.1016/j.ic.2015.03.011

[18] Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. 2017. From model checking to equilibrium checking: Reactive modules for rational verification. *Artif. Intell.* 248 (2017), 123–157. https://doi.org/10.1016/j.artint.2017.04.003

[19] Julian Gutierrez, Paul Harrenstein, and Michael J. Wooldridge. 2017. Reasoning about equilibria in game-like concurrent systems. *Ann. Pure Appl. Log.* 168, 2 (2017), 373–403.

[20] Julian Gutierrez, Sarit Kraus, and Michael J. Wooldridge. 2019. Cooperative Concurrent Games. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019*, Edith Elkind, Manuela Veloso, Noa Agmon, and Matthew E. Taylor (Eds.). International Foundation for Autonomous Agents and Multiagent Systems, 1198–1206. http://dl.acm.org/citation.cfm?id=3331822

[21] Julian Gutierrez, Aniello Murano, Giuseppe Perelli, Sasha Rubin, Thomas Steeples, and Michael Wooldridge. 2020. Equilibria for games with combined qualitative and quantitative objectives. *Acta Informatica* (2020), 1–26. Publisher: Springer.

[22] Julian Gutierrez, Aniello Murano, Giuseppe Perelli, Sasha Rubin, and Michael Wooldridge. 2017. Nash equilibria in concurrent games with lexicographic preferences. (2017). Publisher: AAAI.

[23] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael Wooldridge. 2020. Automated Temporal Equilibrium Analysis: Verification and Synthesis of Multi-Player Games. arXiv:2008.05638 [cs.LO]

[24] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael J. Wooldridge. 2018. EVE: A Tool for Temporal Equilibrium Analysis. In *Automated Technology for Verification and Analysis - 16th International Symposium, ATVA 2018, Los Angeles, CA, USA, October 7-10, 2018, Proceedings (Lecture Notes in Computer Science, Vol. 11138)*, Shuvendu K. Lahiri and Chao Wang (Eds.). Springer, 551–557. https://doi.org/10.1007/978-3-030-01090-4_35

[25] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael J. Wooldridge. 2019. On Computational Tractability for Rational Verification. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 329–335. https://doi.org/10.24963/ijcai.2019/47

[26] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, and Michael J. Wooldridge. 2020. Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. *Artif. Intell.* 287 (2020), 103353.

[27] Julian Gutierrez, Giuseppe Perelli, and Michael J. Wooldridge. 2018. Imperfect information in Reactive Modules games. *Inf. Comput.* 261, Part (2018), 650–675. https://doi.org/10.1016/j.ic.2018.02.023

[28] Julian Gutierrez, Giuseppe Perelli, and Michael J. Wooldridge. 2021. Multi-player games with LDL goals over finite traces. *Inf. Comput.* 276 (2021), 104555.

[29] Thomas A. Henzinger. 2005. Games in system design and verification. In *Proceedings of the 10th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2005), Singapore, June 10-12, 2005*, Ron van der Meyden (Ed.). National University of Singapore, 1–4. https://dl.acm.org/citation.cfm?id=1089935

[30] Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA (The IBM Research Symposia Series)*, Raymond E. Miller and James W. Thatcher (Eds.). Plenum Press, New York, 85–103. https://doi.org/10.1007/978-1-4684-2001-2_9

[31] Richard M Karp. 1978. A characterization of the minimum cycle mean in a digraph. *Discrete mathematics* 23, 3 (1978), 309–311.

[32] Eryk Kopczynski. 2006. Half-Positional Determinacy of Infinite Games. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 4052)*, Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener (Eds.). Springer, 336–347. https://doi.org/10.1007/11787006_29

[33] John Nash. 1951. Non-cooperative games. *Annals of mathematics* (1951), 286–295.

[34] John F Nash et al. 1950. Equilibrium points in n-person games. *Proceedings of the national academy of sciences* 36, 1 (1950), 48–49.

[35] Martin J Osborne and Ariel Rubinstein. 1994. *A course in game theory*. MIT press.

[36] Amir Pnueli. 1977. The Temporal Logic of Programs. In *18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA*. IEEE Computer Society, 46–57. https://doi.org/10.1109/SFCS.1977.32

[37] Amir Pnueli and Roni Rosner. 1989. On the Synthesis of an Asynchronous Reactive Module. In *Automata, Languages and Programming, 16th International Colloquium, ICALP89, Stresa, Italy, Proceedings (LNCS, Vol. 372)*. Springer, 652–671. https://doi.org/10.1007/BFb0035790

[38] A. Prasad Sistla and Edmund M. Clarke. 1985. The Complexity of Propositional Linear Temporal Logics. *J. ACM* 32, 3 (1985), 733–749. https://doi.org/10.1145/3828.3837

[39] Larry J. Stockmeyer and Ashok K. Chandra. 1979. Provably Difficult Combinatorial Games. *SIAM J. Comput.* 8, 2 (1979), 151–174. https://doi.org/10.1137/0208013

[40] Alexis Toumi, Julian Gutierrez, and Michael J. Wooldridge. 2015. A Tool for the Automated Verification of Nash Equilibria in Concurrent Games. In *Theoretical Aspects of Computing - ICTAC 2015 - 12th International Colloquium Cali, Colombia, October 29-31, 2015, Proceedings (Lecture Notes in Computer Science, Vol. 9399)*, Martin Leucker, Camilo Rueda, and Frank D. Valencia (Eds.). Springer, 583–594. https://doi.org/10.1007/978-3-319-25150-9_34

[41] Michael Ummels and Dominik Wojtczak. 2011. The Complexity of Nash Equilibria in Limit-Average Games. *CoRR* abs/1109.6220 (2011). http://arxiv.org/abs/1109.6220

[42] Wiebe van der Hoek, Alessio Lomuscio, and Michael J. Wooldridge. 2006. On the complexity of practical ATL model checking. In *5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, May 8-12, 2006*, Hideyuki Nakashima, Michael P. Wellman, Gerhard Weiss, and Peter Stone (Eds.). ACM, 201–208. https://doi.org/10.1145/1160633.1160665

[43] Yaron Velner, Krishnendu Chatterjee, Laurent Doyen, Thomas A. Henzinger, Alexander Moshe Rabinovich, and Jean-François Raskin. 2015. The complexity of multi-mean-payoff and multi-energy games. *Inf. Comput.* 241 (2015), 177–196. https://doi.org/10.1016/j.ic.2015.03.001

[44] Michael J. Wooldridge, Julian Gutierrez, Paul Harrenstein, Enrico Marchioni, Giuseppe Perelli, and Alexis Toumi. 2016. Rational Verification: From Model Checking to Equilibrium Checking. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, Arizona, USA*. AAAI Press, 4184–4191.

[45] Uri Zwick and Mike Paterson. 1996. The complexity of mean payoff games on graphs. *Theoretical Computer Science* 158, 1-2 (1996), 343–359.