Gutierrez, Julian; Steeples, Thomas; Wooldridge, Michael J.

Article

Mean-payoff games with ω-regular specifications

Games

Provided in Cooperation with:
MDPI – Multidisciplinary Digital Publishing Institute, Basel

Suggested Citation: Gutierrez, Julian; Steeples, Thomas; Wooldridge, Michael J. (2022) : Mean-payoff games with ω-regular specifications, Games, ISSN 2073-4336, MDPI, Basel, Vol. 13, Iss. 1, pp. 1-37, https://doi.org/10.3390/g13010019

games

MDPI

Article

Mean-Payoff Games with ω-Regular Specifications

Julian Gutierrez [1,*], Thomas Steeples [2,*] and Michael Wooldridge [2]

[1] Monash Department of Data Science & AI, Monash Data Futures Institute, Green Chemical Futures Building, 13 Rainforest Walk, Monash University, Clayton, VIC 3800, Australia
[2] Department of Computer Science, University of Oxford, Wolfson Building, Parks Road, Oxford OX1 3QD, UK; mjw@cs.ox.ac.uk
* Correspondence: julian.gutierrez@monash.edu (J.G.); thomas.steeples@cs.ox.ac.uk (T.S.)

Abstract: Multi-player mean-payoff games are a natural formalism for modelling the behaviour of concurrent and multi-agent systems with self-interested players. Players in such a game traverse a graph, while attempting to maximise a (mean-)payoff function that depends on the play 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 address this issue, we consider ω-regular specifications. These offer a concise and intuitive way of specifying system behaviours with a comparatively low computational overhead. 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.

Keywords: multi-player games; mean-payoff games; automated verification; temporal logic; game theory; equilibria; multi-agent systems

Citation: Gutierrez, J.; Steeples, T.; Wooldridge, M. Mean-Payoff Games with ω-Regular Specifications. *Games* **2022**, *13*, 19. https://doi.org/10.3390/g13010019

Academic Editor: Ulrich Berger

Received: 22 November 2021
Accepted: 27 January 2022
Published: 9 February 2022

1. Introduction

Modelling concurrent and multi-agent systems such 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]. One widely adopted semantic framework for modelling such systems is that of concurrent game structures [1]. 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 introduce additional structure to represent each player's preferences over the possible paths of the system. Several approaches have been proposed for this purpose. One very natural method involves assigning a weight to every state of the game, and then considering each player's *mean-payoff* over generated paths: a player prefers paths that maximise their mean-payoff [4–6]. 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 [7]. This approach is (relatively) computationally tractable [5], expressive enough to capture applications of interest, and has been receiving increasing attention recently [8]. As such, equilibria for multi-player games with mean-payoff objectives have been studied, and the computation of Nash equilibria in such games shown to be NP-complete [5].

However, it is well-known in the game theory literature that equilibria may have undesirable properties. In the context of our setting, for example, an equilibrium may visit dangerous system states, or lead to a deadlock. Thus, one may also want to check

if there exist equilibria which satisfy some additional desirable computational properties associated with the game. This decision problem—that is, determining whether a given formal specification is satisfied on some (or every) equilibrium of a given multi-agent system modelled as a multi-player game—is known as *Rational Verification* [9,10].

Previous approaches to rational verification have borrowed their methodology from temporal logic model checking, appealing to logics such as Linear Temporal Logic (LTL) [11] and Computation Tree Logic (CTL) [12]. However, since rational verification subsumes automated synthesis, the use of temporal logic specifications introduces high computational complexity [13]. To mitigate this problem, one might use fragments of temporal logic with lower complexity (e.g., GR(1) (generalised Reactivity(1)) formulae [14,15]), but in this work we adopt a different approach. Taking inspiration from automata theory, and in particular from [16], 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 is the case with temporal logic specifications.

In this paper, we offer the following main contributions. We begin by introducing a language for ω-regular specifications and demonstrate that they form a natural construct for representing properties of concurrent games. In Section 3, we study multi-player mean-payoff games with ω-regular specifications in the non-cooperative setting [7], and consider the natural decision problems relating to these games and their Nash equilibria. Following this, in Section 4 we consider a cooperative solution concept derived from the core [7,17]. Finally, in Section 5 we look at reactive module games [18] as a way of succinctly representing systems, and investigate how the use of this representation affects our complexity results. We conclude with a discussion of related work in Section 6, before offering a glossary of terms, acronyms and notation used within the paper.

2. Models, Games, and Specifications

2.1. Games

A concurrent game structure [1] 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;
- For each $i \in \mathrm{Ag}$, Ac_i is a set of actions available to agent 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 nth 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 *path*. Typically, we index paths with square brackets, i.e., the kth state visited in the path π is denoted $\pi[k]$, and we also use *slice* notation to denote prefix, suffixes and fragments of paths. 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 path 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. Furthermore, unless all states are visited infinitely often, there will also exist some set of states that are visited only finitely often. Thus, given a path π, we can define the following two sets, which one can use to define objectives over paths: $\mathrm{Inf}(\pi) = \{s \in \mathrm{St} \mid \pi \text{ visits } s \text{ infinitely often}\}$ and its complement $\mathrm{Fin}(\pi) = \mathrm{St} \setminus \mathrm{Inf}(\pi)$.

2.2. Strategies

In order to describe how each player plays the game, we need to introduce the concept of a strategy. Mathematically, a strategy for a given player i can be understood as a function, $\sigma_i : \mathrm{St}^+ \to \mathrm{Ac}_i$, which maps sequences, or histories, of states into a chosen action for player i. 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 $\Sigma_1 \times \cdots \times \Sigma_{i-1} \times \Sigma_{i+1} \times \cdots \times \Sigma_{|\mathrm{Ag}|}$. 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 path, which, with a small abuse of notation, we will denote by $\pi(\vec{\sigma}, s) : \mathbb{N} \to \mathrm{St}$, 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 paths 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 infinitely often. Typically, we are interested in paths that begin in the game's start state, s^0, and we write $\pi(\vec{\sigma})$ as shorthand for the infinite path $\pi(\vec{\sigma}, s^0)$.

When considering computational questions surrounding concurrent games, it is useful to work with a *finite* representation of strategies. We use 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. Now, it is not hard to see that, if we have multiple finite-memory strategies playing against each other, then the play they generate will be *periodic*: the play must eventually revisit some configuration, and at this point, the game will start to repeat itself. More precisely, any play generated by a collection of finite-memory strategies will consist of a finite non-repeating sequence of states, followed by a finite sequence that repeats infinitely often. Because such a play will be periodic, we can write the path 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. Thus, memoryless strategies can be expressed as functions $\sigma_i : \mathrm{St} \to \mathrm{Ac}_i$, which simply map states to actions; such functions can be directly implemented as lookup tables, in space $O(|\mathrm{St}|)$. 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.

2.3. Mean-Payoff Games

A two-player, mean-payoff game [4,6], is defined by a tuple:

$$G = (V_1, V_2, v^0, E, w).$$

Here V_1 and V_2 are disjoint, with $v^0 \in V_1$. Additionally, we have: $E \subseteq (V_1 \times V_2) \cup (V_2 \times V_1)$ and w is a function with signature $w : E \to \mathbb{Z}$. There are two players, 1 and 2, and the vertices in V_i should be thought of as those that player i controls—informally, the players take turns choosing edges to traverse, each of which is weighted. Player 1 is trying to maximise the average of the weights encountered, whilst player 2 is trying to minimise it. Formally, the game begins in state $v^0 \in V_1$ and player 1 chooses an edge $(v^0, v^1) \in E \cap (V_1 \times V_2)$. Then player 2 chooses an edge $(v^1, v^2) \in E \cap (V_2 \times V_1)$ and this process repeats indefinitely. This induces a sequence of weights, $\vec{w} = w((v^0, v^1)), w((v^1, v^2)), \ldots,$

and player 1 (respectively, 2) chooses edges to try and maximise (respectively, minimise) the *mean-payoff* of $\vec{w}$, denoted $\mathsf{mp}(\vec{w})$, where for $\beta \in \mathbb{Z}^\omega$, we have:

$$\mathsf{mp}(\beta) = \liminf_{n\to\infty} \frac{1}{n} \sum_{i=0}^{n-1} \beta_i.$$

There are two keys facts about two-player, mean-payoff games that we shall use without comment throughout. The first is that memoryless strategies suffice for both players to act optimally (i.e., achieve their maximum payoff) [4]. The second is that every game has a *value* (i.e., a payoff that player 1 can achieve regardless of what player 2 plays) and determining if a game's value is equal to v lies in NP $\cap$ co-NP [6]. In particular, given a game, determining its value can be seen as a problem that lies within TFNP [19].

Extending two-player, mean-payoff games to multiple players, a multi-player mean-payoff game [5] 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. In a multi-player mean-payoff game, a path $\pi = s^0 s^1 \dots$ induces an infinite sequence of weights for each player, $w_i(s^0)w_i(s^1)\dots$ We denote this sequence by $w_i(\pi)$. Under a given path, π, a player's payoff is given by $\mathsf{mp}(w_i(\pi))$. For notational convenience, we will write $\mathsf{pay}_i(\pi)$ for $\mathsf{mp}(w_i(\pi))$. We can then define a preference relation over paths 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 $\pi' \succeq_i \pi$. Note that, since strategy profiles $\vec{\sigma}$ induce unique plays $\pi(\vec{\sigma})$, we can lift preference relations from plays to strategy profiles, for example writing $\vec{\sigma_1} \succeq_i \vec{\sigma_2}$ as a shorthand for $\pi(\vec{\sigma_1}) \succeq_i \pi(\vec{\sigma_2})$.

In what follows, we refer to multi-player, mean-payoff games simply as mean-payoff games, and refer to two-player, mean-payoff games explicitly as such.

2.4. Solution Concepts

To analyse our games, we make use of solution concepts from both the *non-cooperative* and *cooperative* game theory literatures. With respect to non-cooperative solution concepts, a strategy profile $\vec{\sigma}$ is said to be a *Nash equilibrium* [20,21] 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. In addition to Nash equilibrium, we also consider the cooperative solution concept known as the *core* [17,22]. 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*: a strategy profile that is stable under multilateral deviations, assuming the remainder of the players 'stay put' [23,24]. We will not use strong Nash equilibria in this work, and only mention the concept in order to emphasise how strong Nash equilibria are different from the core.

2.5. ω-Regular Specifications

In [16], 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 path to *model* a specification. Let π be a path, 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 both $\pi \models \alpha$ and $\pi \models \beta$.

Note that we use Inf in two different, but interrelated senses. First, we use it as an operator over paths, as in $\mathrm{Inf}(\pi)$, to denote the set of states visited infinitely often in a path 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 follows:

Type	**Associated Sets**	**ω-Regular Specification**
Büchi	$F \subseteq \mathrm{St}$	$\mathrm{Inf}(F)$
Co-Büchi	$G \subseteq \mathrm{St}$	$\mathrm{Fin}(G)$
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	$(F_k)_{k\in K} \subseteq 2^{\mathrm{St}}$	$\bigvee_{k\in K} \mathrm{Inf}(F_k) \wedge \mathrm{Fin}(\overline{F_k})$

Our ω-regular specifications are equivalent to Emerson-Lei conditions [25], albeit with a different syntax. We can in fact represent all possible ω-regular winning conditions—as another example, consider parity conditions. Suppose each state is labelled by a function, $\mu : \mathrm{St} \to \mathbb{N}$, with $\mu(s) \leq m$ for some $m \in \mathbb{N}$, for all $s \in \mathrm{St}$. Given a path, π, the traditional parity condition is satisfied when $\min\{\mu(s) \mid s \in \mathrm{Inf}(\pi)\}$ is odd. The sets of interest are defined by:

$$F_k = \{s \in \mathrm{St} \mid \mu(s) = k\}$$

Then, assuming m is odd (the formula for m even is similar), the parity condition can be expressed by the following specification:

$$\alpha = \mathrm{Fin}(F_0) \wedge (\mathrm{Inf}(F_1) \vee (\mathrm{Fin}(F_2) \wedge (\mathrm{Inf}(F_3) \vee \cdots \vee (\mathrm{Fin}(F_{m-1}) \wedge \mathrm{Inf}(F_m)))))$$

With ω-regular specifications defined, we can talk about them in the context of games. Let $\vec{\sigma}$ be some strategy profile. Then, $\vec{\sigma}$ induces some path, $\pi(\vec{\sigma})$, and given that ω-regular specifications are defined on paths, we can talk about paths induced by strategies modelling specifications. However, we are not interested in whether the paths induced by arbitrary strategies model a given specification—it is more natural in the context of *multi-player games* to ask whether the paths 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, we are interested in the Nash equilibria and the core, and whether the paths induced by strategy profiles that form an instance of these solution concepts model a specification.

Example 1. *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*. Furthermore, 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.*

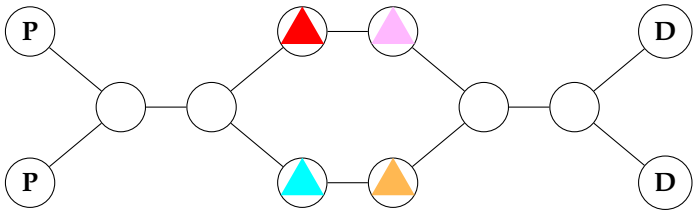

Figure 1. Robots manoeuvering in a warehouse.

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—a socially undesirable outcome.

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 robot i visits a pickup point P_i and similarly label the set of delivery points D_i, we can express this condition concisely with the following ω-regular specification:

$$\bigwedge_{i \in [4]} Inf(P_i) \wedge Inf(D_i). \tag{1}$$

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.

2.6. Mean-Payoff Games with ω-Regular Specifications

Given that we have proposed ω-regular specifications as an alternative to LTL [11] specifications, it is natural to ask how they compare. The connection between them is given by the following statement:

Proposition 1. *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 paths π, we have $\pi \models \alpha$ if and only if $\lambda(\pi) \models \varphi$.*

Proof. Without loss of generality, we may assume that α is written in conjunctive normal form, that is:

$$\alpha = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} C_{i,j} \right),$$

where each $C_{i,j}$ is an atom of the form $\mathrm{Inf}(F)$ or $\mathrm{Fin}(F)$ for some subset $F \subseteq \mathrm{St}$. We start by introducing a propositional variable p_F for every subset $F \subset \mathrm{St}$. Then, for a given state $s \in \mathrm{St}$, we define:

$$\lambda(s) = \{p_F \in \mathrm{AP} \mid s \in F\}.$$

Then, we simply define:

$$\varphi = \bigwedge_{i=1}^{n} \left(\bigvee_{j=1}^{m} D_{i,j} \right),$$

where $D_{i,j} = \mathbf{G}\,\mathbf{F}\,p_F$ if $C_{i,j} = \mathrm{Inf}(F)$ and $D_{i,j} = \mathbf{F}\,\mathbf{G}\,\neg p_F$ if $C_{i,j} = \mathrm{Fin}(F)$. We claim that for all paths π, we have $\pi \models \alpha$ if and only if $\lambda(\pi) \models \varphi$.

First suppose that $\pi \models \alpha$. Thus, by definition, we have for all $1 \leq i \leq n$ that $\pi \models \bigvee_{j=1}^{m} C_{i,j}$. This in turn implies that there exists some j such that $\pi \models C_{i,j}$. If $C_{i,j} = \text{Inf}(F)$, then this implies that $\text{Inf}(\pi) \cap F \neq \emptyset$. Take any $s \in \text{Inf}(\pi) \cap F$. By definition, we have $p_F \in \lambda(s)$ and so we also have $\lambda(\pi) \models \mathbf{G}\,\mathbf{F}\,p_F$. However, by construction, this implies $\lambda(\pi) \models D_{i,j}$.

Similarly, if $C_{i,j} = \text{Fin}(F)$, then we have $\text{Inf}(\pi) \cap F = \emptyset$. Thus, for all $s \in \text{Inf}(\pi)$, we have $p_F \notin \lambda(s)$ and so we have $\lambda(\pi) \models \mathbf{F}\,\mathbf{G}\,\neg p_F$. By construction, this implies that $\lambda(\pi) \models D_{i,j}$. Thus, for all i, there exists some j such that $\lambda(\pi) \models D_{i,j}$. This implies that $\lambda(\pi) \models \varphi$.

Conversely, suppose that $\lambda(\pi) \models \varphi$. So for all i, there exists a j such that $\lambda(\pi) \models D_{i,j}$. If $D_{i,j} = \mathbf{G}\,\mathbf{F}\,p_F$, then π visits some state $s \in F$ infinitely often. Thus, $\text{Inf}(\pi) \cap F \neq \emptyset$, so $\pi \models \text{Inf}(F)$. Similarly, if $D_{i,j} = \mathbf{F}\,\mathbf{G}\,\neg p_F$, then $\pi \models \text{Fin}(F)$. Either way, we have $\pi \models C_{i,j}$. So for all i, there exists some j such that $\pi \models C_{i,j}$, implying that $\pi \models \alpha$. □

Thus, ω-regular specifications can be seen as being 'isomorphic' to a strict subset of LTL (it is straightforward to come up with LTL formulae that cannot be written as ω-regular specifications—take for instance $\mathbf{G}\,\varphi$, where φ is some non-trivial propositional formula). As 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 [10,17], 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 path models the specification. Secondly, given a game and an ω-regular specification, we can ask if the specification is modelled by the path/paths 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 $\vec{\sigma} \in \text{NE}(G)$ and $\pi(\vec{\sigma}) \models \alpha$?

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

A natural dual to E-NASH is the A-NASH problem, which instead of asking if the specification holds in the path induced by *some* Nash equilibrium, asks if the specification holds in *all* equilibria:

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

These decision problems were first studied in the context of iterated Boolean games [26], and are the 'flagship' decision problems of rational verification [18].

In the cooperative setting, the analogous decision problems are defined by substituting $\text{CORE}(G)$ for $\text{NE}(G)$. We refer to these problems as MEMBERSHIP, E-CORE, and A-CORE, respectively, (with a small abuse of notation for the first problem: context will make it clear whether we are referring to the non-cooperative or cooperative problem). These variants were first studied in the setting of LTL games [17].

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.

Before we proceed to studying all these problems in detail, we note that even though some other types of games (for example, two-player, turn-based, zero-sum mean-payoff, or parity games) can be solved only using memoryless strategies, this is not the case in our setting:

Proposition 2. *There exist games G and ω-regular specifications α such that $\pi(\vec{\sigma}) \models \alpha$ for some Nash equilibrium $\vec{\sigma}$, but for which there exists no memoryless Nash equilibrium $\vec{\sigma}$ such that $\pi(\vec{\sigma}) \models \alpha$. The statement also holds true for the core.*

Proof. Consider the game, $G = (M, (w_i)_{i \in \mathrm{Ag}})$, where $M = (\mathrm{Ag}, \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i \in \mathrm{Ag}}, \mathrm{tr})$ is defined as follows: $\mathrm{Ag} = \{1, 2\}$, $\mathrm{St} = \{\mathrm{mid}, \mathrm{right}, \mathrm{left}\}$, $\mathrm{Ac}_1 = \mathrm{Ac}_2 = \{\mathbf{R}, \mathbf{L}\}$, with the transition relation defined to be,

St \ Ac	(R, R)	(R, L)	(L, R)	(L, L)
mid	right	mid	mid	left
right	right	mid	mid	mid
left	mid	mid	mid	left

and the weight functions for the players defined as,

$w_i(s)$	1	2
mid	0	0
right	1	0
left	0	1

Thus, the game looks like the following concurrent game structure (Figure 2):

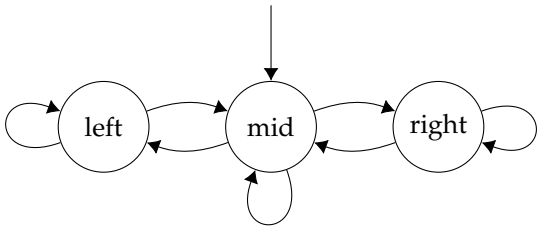

Figure 2. A game where a finite-memory Nash equilibrium is required to model an ω-regular specification.

In this game, each player decides whether they want to go left or right. When they both agree on what direction they want to go, they move in that direction. If they disagree, then they end up in the middle state. We now produce a finite memory strategy profile, $\vec{\sigma}$, and a specification α such that $\vec{\sigma}$ is a Nash equilibrium and $\pi(\vec{\sigma}) \models \alpha$. The basic idea is that with finite memory strategies, the two players can alternate between the left and right state, each achieving a strictly positive payoff, threatening to punish one another if either deviates from this arrangement, whilst this is not possible in memoryless strategies, as in this case the players would have to take the same action each time in the middle state.

Consider the following strategy, σ_1, for player 1 (Figure 3):

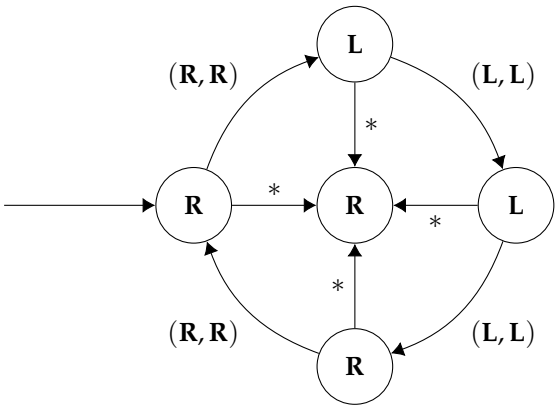

Figure 3. A finite memory strategy for player 1 which forms part of a Nash equilibrium which models the given specification. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

Furthermore, the corresponding strategy, σ_2 for player 2 (Figure 4):

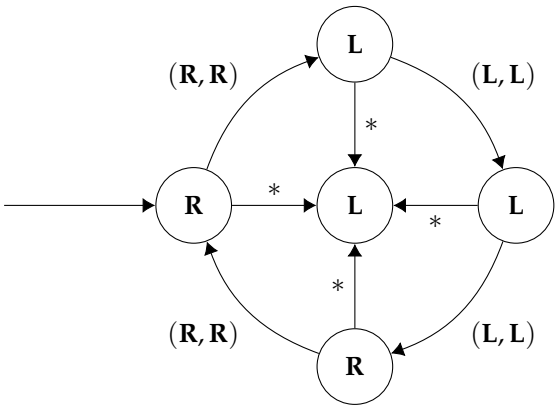

Figure 4. A finite memory strategy for player 2 which forms part of a Nash equilibrium which models the given specification. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

It is easy to verify that $\vec{\sigma} = (\sigma_1, \sigma_2)$ is a Nash equilibrium. Each player gets a mean payoff of $1/4$ under $\vec{\sigma}$. Suppose player 1 deviates to another strategy, which does not match the sequence of actions as dictated by σ_2. Then player 2 will just start playing **L** forever, meaning that the game will never enter the right state again, implying player 1 gets a payoff of zero. By symmetry, we can conclude that this is a Nash equilibrium. Moreover, letting α be the ω-regular specification $\alpha = \mathrm{Inf}(\{\mathrm{right}\}) \wedge \mathrm{Inf}(\{\mathrm{left}\})$, we see that $\pi(\vec{\sigma}) \models \alpha$.

However, note that there cannot exist a memoryless Nash equilibrium, $\vec{\sigma}$, such that $\pi(\vec{\sigma}) \models \alpha$. In a memoryless strategy, for a given state, both players must commit to an action and only play that given action vector in that state. If both players play **R** in the middle state, then they will never be able to reach the left state, and if they both play **L** in the middle state, they will never reach the right state. Additionally, if they disagree, then they will perpetually stay in the middle state. Thus, there cannot exist a memoryless strategy profile $\vec{\sigma}$, such that $\pi(\vec{\sigma}) \models \alpha$. This example demonstrates that, in general, memoryless strategies are not powerful enough for our games: there may exist a Nash equilibrium which satisfies the specification, with no memoryless Nash equilibrium which satisfies it. Finally, note that $\vec{\sigma}$ is also in the core—individual deviations have been already accounted for, and clearly no group deviation containing both players strictly improves both pay-offs. □

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 and even remains NP-hard when restricted to memoryless strategies—thus, no worse than solving a multi-player mean-payoff game [5]. Because A-NASH is the dual problem of E-NASH, it follows that A-NASH is co-NP-complete. In order to obtain some of these results, we also provide a semantic characterisation of the paths 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—arguably, the simplest, yet still computationally important, model of strategies one may want to consider for multi-agent systems.

Theorem 1. *For memoryless strategies,* MEMBERSHIP *is in* P.

Proof. We verify that a given strategy profile is a Nash equilibrium in the following way. We begin by calculating the payoff of each player in the strategy profile by 'running' the strategy and keeping note of the game states. When we encounter the same game state twice, we can simply take the average of the weights encountered at the states between the two occurrences, and that will be the payoff for a given player. By the pidgeonhole principle, it will take no more than $|\mathrm{St}| + 1$ time steps to get to this point, and thus, this can be done in linear time.

Once we have each player's payoff, we can determine if they have any incentive to deviate in the following way—for each player, look at the graph induced by the strategy profile excluding them. Formally, the graph induced by a partial strategy profile $\vec{\sigma}_{-i}$, denoted by $G[\vec{\sigma}_{-i}] = (V, E)$ is defined as follows: the set of nodes, V, simply consists of the set of states of the game, that is, $V = \mathrm{St}$ and the set of edges are simply the moves available to player i—that is, if $e = (s^1, s^2) \in E$, then there exists some $\mathrm{ac}'_i \in \mathrm{Ac}_i$ such that $\mathrm{tr}(s^1, (\vec{\sigma}_{-i}(s^1), \mathrm{ac}'_i)) = s^2$.

We can use Karp's algorithm [27] to determine the maximum payoff that this player can achieve given the other players' strategies. If this payoff is higher than the payoff originally achieved, then this means player i can successfully deviate from the given strategy profile and be better off, and so it is not a Nash equilibrium. If we do this for each player, and the maximum payoff each player can achieve to equal to their payoff in the given strategy profile, then we can conclude it is a Nash equilibrium, and moreover, we have determined this in polynomial time.

To determine if the strategy profile satisfies the specification, we run the strategy as before, and determine the periodic path which the game will end up in. This tells us which states will be finitely and infinitely visited, which in turn induces a valuation which will either model or not model the specification. Checking this can be done in polynomial time and thus, we can conclude that for memoryless strategies, MEMBERSHIP lies in P. □

The simplicity of the above algorithm may raises hopes that it might extend to finite-memory strategies. However, in this case, the 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. Now whilst there is an exponential dependency on the number of players in the input to the problem, this bound on the number of configurations is not necessarily polynomial in the size of the input. More precisely, the size of the underlying concurrent game structure is $|\mathrm{St}| \cdot |\mathrm{Ac}|^{|\mathrm{Ag}|}$ and so if $|Q|$ is larger that $|\mathrm{Ac}|$, the number of configurations will grow exponentially faster than the size of the input. Thus, we cannot use the above algorithm in the case of finite memory strategies to get a polynomial time upper bound.

We now consider the E-NASH problem. Instead of providing the full NP-completeness result here, we start by showing that the problem is NP-hard, even for memoryless strate-

gies, 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 [28,29]—a similar, but simpler, argument can be found in [5].

Proposition 3. E-NASH *is* NP-hard, *even for games with one player, constant weights, and memoryless strategies.*

Proof. Let $G = (V, E)$ be a graph with $|V| = n$. We form a mean-payoff game G by letting $\mathrm{Ag} = \{1\}$ and $\mathrm{St} = V$. We pick the initial state arbitrary and label it s^0, and the actions for player 1 correspond to the edges of G. That is, we have $\mathrm{Ac}_1 = E$ and we have $\mathrm{tr}(u, e) = v$ if and only if $e = (u, v) \in E$. Finally, we fix an integer constant $k \in \mathbb{Z}$ and let $w_1(s) = k$ for all $s \in \mathrm{St}$.

Let $F_s = \{s\}$ for each $s \in \mathrm{St}$ and let α be the following specification:

$$\alpha = \bigwedge_{s \in \mathrm{St}} \mathrm{Inf}(F_s).$$

We claim that G has a Hamiltonian cycle if and only if G has a memoryless Nash equilibrium $\vec{\sigma}$ such that $\pi(\vec{\sigma}) \models \alpha$.

First suppose that G has a Hamiltonian cycle, $\pi = v^0 v^1 \dots v^{n-1}$. We define a memoryless strategy, σ_1, for player 1 by setting $\sigma_1(v^i) = v^{i+1}$, where the superscript is interpreted modulo n. As π is a Hamiltonian cycle, it visits every node, so σ_1 is a well-defined, total function. By definition, we see that $\pi(\sigma_1) = \pi$, so we have $\pi(\sigma) \models \alpha$. Additionally, as each state has the same payoff, this strategy is trivially a Nash equilibrium.

Now suppose that G has a memoryless Nash equilibrium σ_1 such that $\pi(\sigma_1) \models \alpha$. Let $\pi(\sigma_1) = \pi$. Since σ_1 is memoryless, π must be of the form $\pi = \pi[..i]\pi[i..j]^\omega$ for integers i and j. Without loss of generality, assume that i and j are the smallest integers such that this holds. Moreover, since $\pi \models \alpha$, the path π visits every state infinitely often so we must have that $\pi[i..j]$ contains every state, and by memorylessness, it must be a cycle. Thus, by definition, $\pi[i..j]$ is a Hamiltonian cycle. □

Theorem 1 and Proposition 3 together establish NP-completeness 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 we shall show later, the problem is also NP-complete in the general case. To prove this, we need to develop an alternative characterisation of Nash equilibria.

To do this, we need to introduce the notion of the *punishment value* in a multi-player mean-payoff game; cf., [30,31]. 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:

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

How efficiently can we calculate this value? As established in [5], 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 [4]. 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 [27] 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 [5,15].

Lemma 1. *Let π be a path in G and let $\{\vec{ac}[k]\}_{k\in\mathbb{N}}$ be the path of associated action profiles. Then there is a Nash equilibrium, $\vec{\sigma} \in NE(G)$, such that $\pi = \pi(\vec{\sigma})$ 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 $z_i \leq \mathsf{pay}_i(\pi)$.*

Proof. First assume that we have some Nash equilibrium $\vec{\sigma} \in \mathrm{NE}(G)$ such that $\pi = \pi(\vec{\sigma})$. Suppose there does not exist any $z \in \mathbb{Q}^{\mathrm{Ag}}$ with the desired properties. Furthermore, let us first suppose that for all $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, with $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, there exists some player $i \in \mathrm{Ag}$ such that $z_i > \mathsf{mp}(w_i(\pi))$. However, this is true for all $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$. So we have $\mathrm{pun}_i(s^0) > \mathsf{mp}(w_i(\pi))$. This means that player i might as well play the positional strategy which ensures they achieve at least the punishment value. This in turn means that player i has some deviation, contradicting the fact that $\vec{\sigma}$ is a Nash equilibrium.

So instead, it must be the case that for all $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, with $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, there is some time step k, a player i and an action ac'_i such that $\mathrm{pun}_i(\mathrm{tr}(\pi[k], (\vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}'_i))) > z_i$. However, since this is true for all z_i, it is true for $z_i = \max\{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$. Furthermore, since $\mathrm{pun}_i(\mathrm{tr}(\pi[k], (\vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}'_i))) \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, this gives a contradiction—we cannot have $\mathrm{pun}_i(\mathrm{tr}(\pi[k], (\vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}'_i))) > z_i$. Thus, it follows that the first part of the statement is true.

Now assume that there exists some $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$ with the properties as prescribed in the statement of the lemma. We define a strategy profile $\vec{\sigma}$ in the following way. Each player follows π. If any player chooses to deviate from π, say at state $\pi[k]$, with an action ac'_i then the remaining players play the punishing strategy which causes player i to have a payoff of at most $\mathrm{pun}_i(\mathrm{tr}(\pi[k], \vec{\mathrm{ac}}[k]_{-i}, \mathrm{ac}'_i)) \leq z_i \leq \mathsf{pay}_i(\pi)$. Thus, no player has any incentive to deviate away from $\vec{\sigma}$ and so we have a Nash equilibrium with $\pi(\vec{\sigma}) = \pi$. □

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 path π in $G[\vec{z}; F]$ with $z_i \leq \mathsf{pay}_i(\pi)$ for each player i which visits every state infinitely often. Trivially, if this graph is not strongly connected, then no path 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 4. *Let $G = (V, E)$ be a strongly connected graph, let $\{w_i\}_{i\in Ag}$ be a set of weight functions, and let $\vec{z} \in \mathbb{Q}^{Ag}$. Then, we can determine if there is some path π with the properties,*

- *π visits every state infinitely often;*
- *$z_i \leq \mathsf{pay}_i(\pi)$ for each $i \in Ag$,*

in polynomial time.

Conceptually, Proposition 4 is similar to Theorem 18 of [5], but with two key 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 [5] 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 be used not only for verification, but also for synthesis.

For clarity of presentation, we shall split the above proposition into two constituent lemmas. To do this, we begin by defining a system of linear inequalities. We then go on to show that there is a path π with the desired properties if and only if this system has a solution—one lemma for each direction. As the system of inequalities can be determined in polynomial time, this yields our result.

For a graph G, a set of weight functions $\{w_i\}_{i\in \mathrm{Ag}}$ be a set of weight functions, and a vector $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, we define a system of linear inequalities, $\ell(G, \{w_i\}_{i\in \mathrm{Ag}}, \vec{z})$ as follows: for each agent $i \in \mathrm{Ag}$, and each edge $e \in E$, introduce a variable $x_{i,e}$, along with the following inequalities:

(i) $x_{i,e} \geq 0$ for each agent $i \in \mathrm{Ag}$ and for each edge $e \in E$.
(ii) $\sum_{e\in E} x_{i,e} = 1$ for each agent $i \in \mathrm{Ag}$.
(iii) $\sum_{e\in \mathrm{In}(v)} x_{i,e} = \sum_{e\in \mathrm{Out}(v)} x_{i,e}$ for all $i \in \mathrm{Ag}$ and for each $v \in V$.
(iv) $z_i \leq \sum_{e\in E} x_{i,e} \cdot w_i(e)$ for all $i \in \mathrm{Ag}$.
(v) $\sum_{e\in E} x_{i,e} \cdot w_i(e) \leq \sum_{e\in E} x_{j,e} \cdot w_i(e)$ for all $i, j \in \mathrm{Ag}$.

It is worth briefly discussing what this system is actually encoding. Roughly speaking, the set $\mathcal{C}_i = \{x_{i,e}\}_{e\in E}$ defines a cycle for each player that makes sure their payoff is greater than z_i. Each $x_{i,e}$ represents the proportion that a given edge is visited in the cycle. The idea is that we define a path by visiting each $\mathcal{C}_i$ an appropriate number of times, before travelling to the next cycle and visiting that repeatedly.

Conversely, if there exists some path with the stated properties, it will also define a solution to the system of inequalities. This has the corollary that if there is a Nash equilibrium in the game, then there is a finite state Nash equilibrium as well.

In what follows, for an edge $e = (u, v)$ and a weight function $w : V \to \mathbb{Z}$, we define $w(e) := w(u)$. We also extend weight functions to finite paths in the natural way, by summing along them.

Lemma 2. *Let $G = (V, E)$ be a strongly connected graph, let $\{w_i\}_{i\in Ag}$ be a set of weight functions, let $\vec{z} \in \mathbb{Q}^{Ag}$. Furthermore, suppose there exists some path π such that $z_i \leq \mathsf{pay}_i(\pi)$ for each $i \in Ag$, which visits every state infinitely often. Then $\ell(G, \{w_i\}_{i\in Ag}, \vec{z})$ has a solution.*

Proof. First suppose there exists some path π with the stated properties. For each $n > 0$ and $e \in E$, define $\lambda(n, e)$ to be the following quantity:

$$\lambda(n, e) = \frac{\#\{j < n \mid (\pi[j], \pi[j + 1]) \in E\}}{n}.$$

Informally, for a given edge e, $\lambda(n, e)$ gives us the proportion that e appears in the prefix $\pi[..n]$. Note that $0 \leq \lambda(n, e) \leq 1$ for all $e \in E$ and all $n > 0$. Additionally, for each n, it easy to see we have:

$$\sum_{e\in E} \lambda(n, e) = 1. \tag{2}$$

By definition, for each $i \in \mathrm{Ag}$ we have:

$$\mathsf{pay}_i(\pi) = \liminf_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]).$$

Thus, there must exist some subsequence of the natural numbers, $\{n_t^i\}_{t\in\mathbb{N}}$, such that:

$$\mathsf{pay}_i(\pi) = \lim_{t\to\infty} \frac{1}{n_t^i} \sum_{k=0}^{n_t^i-1} w_i(\pi[k]).$$

With this defined, we introduce a sequence of numbers, $\{\varphi_t^i(e)\}_{t\in\mathbb{N}}$ for each edge $e \in E$, by defining $\varphi_t^i(e) = \lambda(n_t^i, e)$. Now, as $\{\varphi_t^i(e)\}_{t\in\mathbb{N}}$ is a bounded sequence, by the Bolzano-Weierstrass theorem, there must be a convergent subsequence for each edge e, $\{\psi_t^i(e)\}_{t\in\mathbb{N}}$. We define $x_{i,e}^* = \lim_{t\to\infty} \psi_t^i(e)$ and claim that these form a solution to the system of inequalities.

Since $\varphi_t^i(e) = \lambda(n_t^i, e)$, it is clear that we have $0 \leq x_{i,e}^* \leq 1$. Thus, Inequality (i) is satisfied. Additionally, by Equation (2), we can deduce that for each t, and for every $i \in Ag$, we have:

$$\sum_{e\in E} \psi_t^i(e) = 1.$$

Taking limits, we can conclude that Inequality (ii) is satisfied. To establish (iii), by definition of λ, fix a $v \in V$. In a path, if we enter a node, we must exit it. Thus, we have:

$$-1 \leq \sum_{e\in \text{In}(v)} n \cdot \lambda(n, e) - \sum_{e\in \text{Out}(v)} n \cdot \lambda(n, e) \leq 1,$$

implying,

$$-\frac{1}{n_t^i} \leq \sum_{e\in \text{In}(v)} \varphi_t^i(e) - \sum_{e\in \text{Out}(v)} \varphi_t^i(e) \leq \frac{1}{n_t^i}.$$

Taking the relevant subsequence and letting $t \to \infty$, we can deduce that Inequality (iii) is satisfied. To establish (iv), first note that for all $i \in$ Ag, we have,

$$\begin{aligned}
\mathsf{pay}_i(\pi) &= \liminf_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]) \\
&= \lim_{t\to\infty} \frac{1}{n_t^i} \sum_{k=0}^{n_t^i-1} w_i(\pi[k]) \\
&= \lim_{t\to\infty} \sum_{e\in E} \lambda(n_t^i, e) \cdot w_i(e) \\
&= \lim_{t\to\infty} \sum_{e\in E} \varphi_t^i(e) \cdot w_i(e) \\
&= \lim_{t\to\infty} \sum_{e\in E} \psi_t^i(e) \cdot w_i(e) \\
&= \sum_{e\in E} x_{i,e} \cdot w_i(e).
\end{aligned}$$

The equality between lines 4 and 5 is valid, as we have established that the limit exists in line 2. Thus, since we have $z_i \leq \mathsf{pay}_i(\pi)$, by assumption, we can conclude that Inequality (iv) is valid. In a similar fashion, note that for all $i, j \in$ Ag, we have,

$$\begin{aligned}
\mathsf{pay}_i(\pi) &= \liminf_{n\to\infty} \frac{1}{n} \sum_{k=0}^{n-1} w_i(\pi[k]) \\
&\leq \liminf_{t\to\infty} \frac{1}{n_t^j} \sum_{k=0}^{n_t^j-1} w_i(\pi[k]) \\
&= \liminf_{t\to\infty} \sum_{e\in E} \lambda(n_t^j, e) \cdot w_i(e) \\
&= \liminf_{t\to\infty} \sum_{e\in E} \varphi_t^j(e) \cdot w_i(e) \\
&\leq \lim_{t\to\infty} \sum_{e\in E} \psi_t^j(e) \cdot w_i(e) \\
&= \sum_{e\in E} x_{j,e} \cdot w_i(e).
\end{aligned}$$

This, together with Inequality (iv), implies that Inequality (v) is satisfied.

Thus, putting all this together, we can conclude that if there exists some path with the stated properties, then there also exists a solution to $\ell(G, \{w_i\}_{i\in \mathrm{Ag}}, \vec{z})$, the system of linear inequalities. □

We now show the other direction:

Lemma 3. *Let $G = (V, E)$ be a strongly connected graph, let $\{w_i\}_{i\in Ag}$ be a set of weight functions, let $\vec{z} \in \mathbb{Q}^{Ag}$. Suppose that $\ell(G, \{w_i\}_{i\in Ag}, \vec{z})$ has a solution. Then there exists some path π such that $z_i \leq \mathsf{pay}_i(\pi)$ for each $i \in Ag$, which visits every state infinitely often.*

Proof. Given that there is a solution to $\ell(G, \{w_i\}_{i\in \mathrm{Ag}}, \vec{z})$, there must exist a solution that consists of rational numbers. Thus, letting $x^*_{i,e}$ be a rational solution, we can write $x^*_{i,e} = p_{i,e}/q$, for some $p_{i,e} \in \mathbb{N}$, and some appropriately chosen $q \in \mathbb{N}$.

Now, for each $i \in \mathrm{Ag}$, we form a multigraph, $G_i = (V_i, E_i)$, which takes G and replaces each edge e by $p_{i,e}$ copies. Note that whilst G is strongly connected, some of the $p_{i,e}$ may be equal to 0, which would mean that G_i is disconnected.

Now, by Inequality (iii), for each $v \in V$, we have:

$$\sum_{e\in \mathrm{In}(v)} p_{i,e} = \sum_{e\in \mathrm{Out}(v)} p_{i,e}.$$

Thus, the in-degree of each vertex in V_i is equal to its out-degree. Thus, each strongly connected component of G_i contains an Eulerian cycle (An Eulerian cycle is a path that starts and ends at the same node which visits every edge exactly once). Interpreting each of these Eulerian cycles as paths in G, we get a set of (not necessarily simple) cycles for each agent, $C^1_i, \ldots C^{m_i}_i$, where $m_i \leq |V|$.

Now for each agent $i \in \mathrm{Ag}$, and every $n \in \mathbb{N}$, we define a cycle $\mathcal{C}^n_i$ which starts at the first state of C^1_i, traces C^1_i n times, takes the shortest path from the start state of C^1_i to C^2_i, traces C^2_i n times and repeats this process for all the cycles of agent i, until it has traced $C^{m_i}_i$ n times. From here, it then takes the shortest path to the start state of C^1_i.

Let M_i be the largest of the absolute values of the weights of agent i. That is, $M_i = \max_{v\in V} |w_i(v)|$. Then we have for all $i, j \in \mathrm{Ag}$,

$$n \cdot \sum_{e\in E} p_{j,e} \cdot w_i(e) - |V|^2 \cdot M_i \leq w_i(\mathcal{C}^n_j) \leq n \cdot \sum_{e\in E} p_{j,e} \cdot w_i(e) + |V|^2 \cdot M_i,$$

and,

$$n \cdot \sum_{e\in E} p_{j,e} \leq |\mathcal{C}^n_j| \leq n \cdot \sum_{e\in E} p_{j,e} + |V|^2.$$

Putting these together, we get:

$$\begin{aligned}
\frac{w_i(\mathcal{C}^n_j)}{|\mathcal{C}^n_j|} &\geq \frac{n \cdot \sum_{e\in E} p_{j,e} \cdot w_i(e) - |V|^2 \cdot M_i}{n \cdot \sum_{e\in E} p_{j,e} + |V|^2} \\
&\geq \frac{n \cdot \sum_{e\in E} p_{i,e} \cdot w_i(e) - |V|^2 \cdot M_i}{n \cdot \sum_{e\in E} p_{j,e} + |V|^2} \\
&= \frac{\sum_{e\in E} x^*_{i,e} \cdot w_i(e) - \frac{|V|^2 \cdot M_i}{n\cdot q}}{\sum_{e\in E} x^*_{j,e} + \frac{|V|^2}{n\cdot q}} \\
&\geq \frac{z_i - \frac{|V|^2 \cdot M_i}{n\cdot q}}{1 + \frac{|V|^2}{n\cdot q}}.
\end{aligned}$$

Taking limits, we see that:

$$\lim_{n\to\infty} \frac{w_i(\mathcal{C}_j^n)}{|\mathcal{C}_j^n|} \geq z_i.$$

Thus, for n_i^* sufficiently large, we must also have, for all $j \in \text{Ag}$:

$$\frac{w_i(\mathcal{C}_j^{n_i^*})}{|\mathcal{C}_j^{n_i^*}|} \geq z_i.$$

Moreover, if n_i^* is large enough, the above equation will be true for all $n \geq n_i^*$. Assuming this is the case, we set $n^* = \max_{i\in\text{Ag}} n_i^*$. Then, for all $i, j \in \text{Ag}$, we have,

$$\frac{w_i(\mathcal{C}_j^{n^*})}{|\mathcal{C}_j^{n^*}|} \geq z_i.$$

We are now ready to define a path π^N, which will form the basis for our path of interest, π. The path starts by visiting $\mathcal{C}_1^{n^*}$ N times, before taking the shortest path to the start of $\mathcal{C}_2^{n^*}$, and visiting that N times, and so on. Once $\mathcal{C}_{|\text{Ag}|}^{n^*}$ has been visited N times, we then take the shortest path that visits any states that have not been yet visited, before returning to the start vertex of $\mathcal{C}_1^{n^*}$.

Let us look at each player's payoff on π^N. We have:

$$\begin{aligned}
\frac{w_i(\pi^N)}{|\pi^N|} &\geq \frac{N \cdot \sum_{j\in\text{Ag}} w_i(\mathcal{C}_j^{n^*}) - |V| \cdot |\text{Ag}| \cdot M_i - |V|^2 \cdot M_i}{N \cdot \sum_{j\in\text{Ag}} |\mathcal{C}_j^{n^*}| + |V| \cdot |\text{Ag}| + |V|^2} \\
&\geq \frac{N \cdot \sum_{j\in\text{Ag}} z_i \cdot |\mathcal{C}_j^{n^*}| - |V| \cdot |\text{Ag}| \cdot M_i - |V|^2 \cdot M_i}{N \cdot \sum_{j\in\text{Ag}} |\mathcal{C}_j^{n^*}| + |V| \cdot |\text{Ag}| + |V|^2} \\
&= \frac{z_i \cdot \sum_{j\in\text{Ag}} |\mathcal{C}_j^{n^*}| - \frac{|V|\cdot|\text{Ag}|\cdot M_i + |V|^2 \cdot M_i}{N}}{\sum_{j\in\text{Ag}} |\mathcal{C}_j^{n^*}| + \frac{|V|\cdot|\text{Ag}| + |V|^2}{N}}.
\end{aligned}$$

Taking limits as before, we can conclude that for N^* sufficiently large, we have:

$$\frac{w_i(\pi^{N^*})}{|\pi^{N^*}|} \geq z_i.$$

We then simply define π to be the path that repeatedly traverses π^{N^*}. It is easy to see that $\mathsf{pay}_i(\pi) \geq z_i$ and that π visits every state infinitely often. □

We can now combine these two results to obtain our proof:

Proof. (Proof of Proposition 4) Given the statement of the proposition, we construct an algorithm which forms the linear program $\ell(G, \{w_i\}_{i\in\text{Ag}}, \vec{z})$ and see if it has a solution. As the linear program is of polynomial size, and as linear programs can be solved in polynomial time, we see that the overall algorithm can be done in polynomial time. By Lemma 3, we see that the algorithm is sound and then by Lemma 2, we see it is complete. □

From the propositions above, we can establish the complexity of E-NASH:

Theorem 2. E-NASH *is* NP-complete.

Proof. For NP-hardness we have Proposition 3. 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 $i \in \text{Ag}$, 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 1 and form the graph $G[\vec{z}; F]$. If it is not strongly connected, then we reject. Otherwise, we use Proposition 4 to determine if the associated linear program has a solution. If it does, then we accept, otherwise we reject. □

Another benefit of splitting Proposition 4 is that it readily yields the following corollary:

Corollary 1. *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 $\vec{\sigma}' \models \alpha$.*

Proof. Suppose G has some Nash equilibrium $\vec{\sigma}$ such that $\vec{\sigma} \models \alpha$. Furthermore, let $\pi = \pi(\vec{\sigma})$. Then by Lemma 1, there exists some $\vec{z} \in \mathbb{Q}^{\mathrm{Ag}}$, with $z_i \in \{\mathrm{pun}_i(s) \mid s \in \mathrm{St}\}$, and a subset $F \subseteq \mathrm{St}$ such that in $G[\vec{z}, F]$, π visits every state infinite often and for each $i \in \mathrm{Ag}$, we have $z_i \leq \mathsf{pay}_i(\pi)$. By Lemma 2, we see that the system of inequalities $\ell(G, \{w_i\}_{i \in \mathrm{Ag}}, \vec{z})$ has a solution. However, then by Lemma 3, we see that there exists some periodic path π' such that in $G[\vec{z}, F]$, again, we have that π' visits every state infinite often and for each $i \in \mathrm{Ag}$, we have $z_i \leq \mathsf{pay}_i(\pi')$. Thus, applying Lemma 1 again, we see that G has some Nash equilibrium $\vec{\sigma}'$ such that $\vec{\sigma}' \models \alpha$ with $\pi' = \pi(\vec{\sigma}')$. Moreover, by the construction in Lemma 3, we see that we may assume that $\vec{\sigma}'$ is a finite-memory strategy. □

4. Cooperative Games

We now move on to investigate cooperative solution concepts, and in particular, the core. We start by studying the relationship between Nash equilibrium and the core in our context. We first establish that they are indeed different:

Proposition 5. *Let G be a game with $|Ag| = 1$. Then, $NE(G) = CORE(G)$. However, if $|Ag| \geq 2$, then there exist games G such that $NE(G) \not\subseteq CORE(G)$, and games G such that $CORE(G) \not\subseteq NE(G)$.*

Thus, the two concepts do not coincide beyond the one-player case. In fact, there are two-player games in which the set of Nash equilibria is empty, while the core is not, which demonstrates that the core is not a refinement of Nash equilibrium. Nor does the other direction hold: Nash equilibrium is not a refinement of the core. The following two games (Figures 5 and 6) serve as witnesses to these claims.

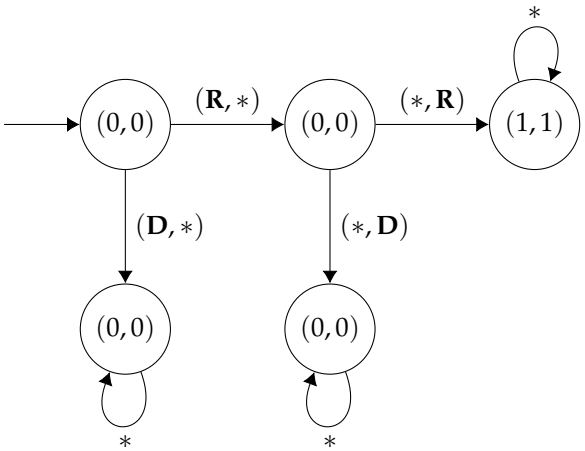

Figure 5. $\mathrm{NE}(G_1) \not\subseteq \mathrm{CORE}(G_1)$. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

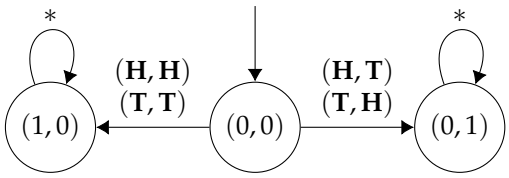

Figure 6. $\mathrm{CORE}(G_2) \not\subseteq \mathrm{NE}(G_2)$. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

As we can see in the game in Figure 5, G_1 has a Nash equilibrium that is not in the core (and in which both players get a mean-payoff of 0—cf., Player 1 choosing **D** at the initial state while Player 2 chooses **D** at the middle state), since the coalition containing both players has a beneficial deviation in which both players get a mean-payoff of 1. On the other hand, in the game in Figure 6, G_2 has a non-empty core (consider every possible memoryless strategy) while it has an empty set of Nash equilibria. Moreover, in both games, the detailed strategies can be implemented without memory.

Regarding memory requirements, as with Nash equilibrium, it may be that, in general, memoryless strategies are not enough to implement all equilibria in the cooperative setting. Indeed, there are games with a non-empty core in which no strategy profile in the core can be implemented in memoryless strategies only. Take, for instance, the game shown in Figure 2, with the weights changed to $w_i(\mathrm{mid}) = 1$, $w_1(\mathrm{right}) = 6 = w_2(\mathrm{left})$. Only if the two players collaborate, and alternate between left and right, they will get their optimal mean-payoffs (of 2). Clearly, such an optimal payoff for both players cannot be obtained using only memoryless strategies.

Another important (game-theoretic) question about cooperative games is whether they have a non-empty core, a property that holds for games with LTL goals and specifications [17]. However, that is not the case for mean-payoff games, at least for games with $|\mathrm{Ag}| > 2$.

Proposition 6. *In mean-payoff games, if $|Ag| \leq 2$, then the core is non-empty. For $|Ag| > 2$, there exist games with an empty core.*

Proof. If $|\mathrm{Ag}| = 1$, because of Proposition 5, the core coincides with the set of Nash equilibria in one-player games, which is always non-empty. For two-player games, let $\vec{\sigma} = (\sigma_1, \sigma_2)$ be any strategy profile. If $\vec{\sigma}$ is not in the core, then either Player 1, or Player 2, or the coalition consisting of both players has a beneficial deviation. If the latter is true, then there is a strategy profile, $\vec{\sigma}' = (\sigma_1', \sigma_2')$ such that $\vec{\sigma}' \succ_i \vec{\sigma}$ for both $i \in \{1, 2\}$. We repeat this process until the coalition of both players does not have a beneficial deviation. This must eventually be the case as each player's payoff is capped by their maximum weight, so either they both reach their corresponding maximum weight, or there comes a point when they cannot beneficially deviate together. At this point, we must either be in the core, or either player 1 or player 2 has a beneficial deviation. If player j ($j \in \{1, 2\}$) has a beneficial deviation, say σ_j, then any strategy profile (σ_j, σ_i), with $i \neq j$, that maximises Player i's mean-payoff is in the core. Thus, for every two-player game, there exists some strategy profile that lies in the core.

However, for three-player mean-payoff games, in general, the core of a game may be empty. Consider the following three-player game G, where each player has two actions, **H**, **T**, and there are four states, P, R, B, Y. The states are weighted for each player as follows:

$w_i(s)$	**1**	**2**	**3**
P	-1	-1	-1
R	2	1	0
B	0	2	1
Y	1	0	2

If the game is in any state other than P, then no matter what set of actions is taken, the game will remain in that state. Thus, we only specify the transitions for the state P:

Ac	St
$(\mathbf{H}, \mathbf{H}, \mathbf{H})$	R
$(\mathbf{H}, \mathbf{H}, \mathbf{T})$	R
$(\mathbf{H}, \mathbf{T}, \mathbf{H})$	B
$(\mathbf{H}, \mathbf{T}, \mathbf{T})$	P
$(\mathbf{T}, \mathbf{H}, \mathbf{H})$	P
$(\mathbf{T}, \mathbf{H}, \mathbf{T})$	Y
$(\mathbf{T}, \mathbf{T}, \mathbf{H})$	B
$(\mathbf{T}, \mathbf{T}, \mathbf{T})$	Y

Thus, the structure of this game looks like the following graph (Figure 7):

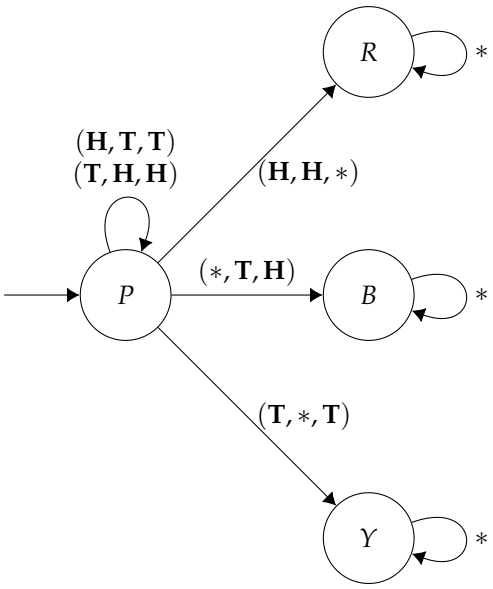

Figure 7. A game with an empty core. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

Note that strategies are characterised by the state that the game eventually ends up in. If the players stay in P forever, then they can all collectively change strategy to move to one of R, B, Y, and each get a better payoff. Now, if the game ends up in R, then players 2 and 3 can deviate by playing $(\mathbf{T}, \mathbf{H})$, and no matter what player 1 plays, they will be in state B, and will be better off. However, similarly, if the game is in B, then players 1 and 3 can deviate by playing $(\mathbf{T}, \mathbf{T})$ to enter state Y, in which they both will be better off, regardless of what player 2 does. Furthermore, finally, if in Y, then players 1 and 2 can deviate by playing $(\mathbf{H}, \mathbf{H})$ to enter R and will be better off regardless of what player 3 plays. Thus, no strategy profile lies in the core. □

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 $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}) \succ_i \pi(\vec{\sigma})$ 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. *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:*

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

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

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

Before we prove this, we need to introduce an auxiliary concept of *two-player, turn-based, zero-sum, multi-mean-payoff games* [32] (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. Paths are defined in the usual way and the payoff of a path 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 [32], 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:

Proof. (Proof of Lemma 4) Let $\vec{\sigma}_{\mathrm{Ag}\setminus C}$ be an arbitrary strategy and let $i \in C$ be an arbitrary agent. Denote $\pi(\vec{\sigma})$ by π and $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$ by π'. We aim to show that $\pi' \succ_i \pi$. Suppose instead it is the case that $\pi \succeq_i \pi'$. Thus, we have $\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$. 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 $\mathsf{mp}(\pi(\vec{\sigma}))$, then $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$ is a winning strategy for player 2 in this game. Now, by [32,33], 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 $\pi(\vec{\sigma}) \succeq_i \pi(\vec{\sigma}'_C, \vec{\sigma}''_{\mathrm{Ag}\setminus C})$. However, this contradicts the assumptions of the lemma, and thus we must have $\pi' \succ_i \pi$ (In [32], their winning condition relates to whether a player's payoff is greater than or equal to a given vector. One can adapt this argument to show it is also true for strict inequalities). □

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:

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

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

Proposition 7. *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 $\pi(\vec{\sigma}) \models \alpha$.*

Proof. Proof follows directly from definitions. □

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.

Theorem 3. *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 $\pi(\vec{\sigma})$, that is, a set of values $z^*_j = \mathsf{pay}_j(\pi(\vec{\sigma}))$ 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 Theorem 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 use a small variation of the construction presented in [34]. Let $P = \{x_1, \ldots, x_n\}$ be a set of atomic propositions. From a Boolean formula $\varphi = \bigwedge_{1 \le c \le m} C_c$ (in conjunctive normal form) over P—where each $C_c = l_{c1} \vee l_{c2} \vee l_{c3}$, and each literal $l_{ck} = x_j$ or $\neg x_j$, with $1 \le k \le 3$, for some $1 \le j \le n$—we construct $M = (\mathrm{Ag}, \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i \in \mathrm{Ag}}, \mathrm{tr})$, an m-player concurrent game structure defined as follows:

- $\mathrm{Ag} = \{1, \ldots, m\}$
- $\mathrm{St} = \{x_v \mid 1 \le v \le n\} \cup \{x'_v \mid 1 \le v \le n\} \cup \{y_0, y_n, y^0, y^*\}$
- $s^0 = y^0$
- $\mathrm{Ac}_i = \{t, f\}$, for every $i \in \mathrm{Ag}$, and $\mathrm{Ac} = \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_m$
- For tr, refer to the figure below, such that $T = \{(t_1, \ldots, t_m)\}$ and $F = \mathrm{Ac} \setminus T$.

The concurrent game structure so generated is illustrated in Figure 8.

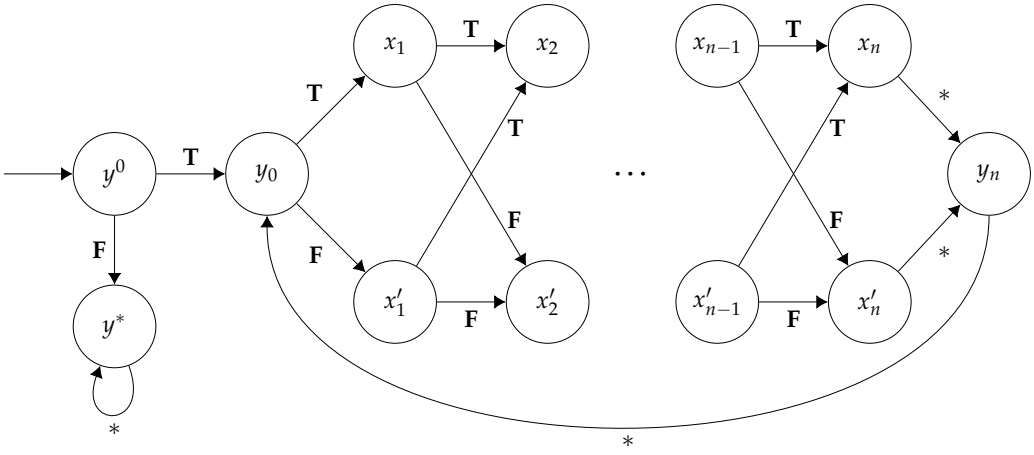

Figure 8. Concurrent game structure for the reduction from 3SAT. The asterisks * are wildcards—they match any action which isn't explicitly detailed on the diagram.

With M at hand, we build a mean-payoff game using the following weight function:

- $w_i(x_v) = 1$ if x_v is a literal in C_i and $w_i(x_v) = 0$ otherwise, for all $i \in \mathrm{Ag}$ and $1 \le v \le n$

- $w_i(x'_v) = 1$ if $\neg x_v$ is a literal in C_i and $w_i(x'_v) = 0$ otherwise, for all $i \in \text{Ag}$ and $1 \leq v \leq n$
- $w_i(y_0) = w_i(y_n) = w_i(y^0) = w_i(y^*) = 0$, for all $i \in \text{Ag}$

Then, we consider the game G over M and any strategy profile (in memoryless strategies) such that $\vec{\sigma}(s^0) = y^*$. For any of such strategy profiles the mean-payoff of every player is 0. However, if φ is satisfiable, then there is a path in M, from y_0 to y_n, such that in such a path, for every player, there is a state in which its payoff is not 0. Thus, the grand coalition Ag has an incentive to deviate since traversing that path infinitely often will give each player a mean-payoff strictly greater than 0. Observe two things. Firstly, that only if the grand coalition Ag agrees, the game can visit y_0 after y^0. Otherwise, the game will necessarily end up in y^* forever after. Secondly, because we are considering memoryless strategies, the path from y_0 to y_n followed at the beginning is the same path that will be followed thereafter, infinitely often. Then, we can conclude that there is a beneficial deviation (necessarily for Ag) if and only if φ is satisfiable, as otherwise at least one of the players in the game will not have an incentive to deviate (because its mean-payoff would continue to be 0). Then, formally, we can conclude that $(G, \sigma) \in$ BEN-DEV if and only if φ is satisfiable. □

From Theorem 3 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 also co-NP-complete.

Theorem 4. *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 $\pi(\vec{\sigma}) \models \alpha$. Thus, we can solve MEMBERSHIP simply by first checking $\pi(\vec{\sigma}) \models \alpha$, which can be done in polynomial time and we reject if that check fails. If $\pi(\vec{\sigma}) \models \alpha$, 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.

Theorem 5. *For memoryless strategies,* E-CORE *is in* Σ_2^{P}.

Proof. Given any instance (G, α), we guess a strategy profile $\vec{\sigma}$ and check that $\pi(\vec{\sigma}) \models \alpha$ 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 $\text{NP}^{\text{co-NP}} = \text{NP}^{\text{NP}} = \Sigma_2^{\text{P}}$. □

From Theorem 5 follows that A-CORE is in Π_2^{P} and, more importantly, that even checking that the core has any memoryless solutions (but not necessarily that it is empty) is also in Σ_2^{P}. This result 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 [17] 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 [7]) simply in the type of beneficial deviation under consideration. Another concept introduced in [17] 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*: let $C \subseteq \text{Ag}$ 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 $\mathsf{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{-C})) \geq z_i$, for every $i \in C$.

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

Lemma 5. *Let π be a path in G. There is $\vec{\sigma} \in CORE(G)$ such that $\pi = \pi(\vec{\sigma})$ 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 $z_i \leq \mathsf{pay}_i(\pi)$.*

Proof. To show the left-to-right direction, suppose that there exists a member of the core $\vec{\sigma} \in \mathrm{CORE}(G)$ with $\pi = \pi(\vec{\sigma})$ and suppose further that there is some coalition $C \subseteq \mathrm{Ag}$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, such that for every $i \in C$ we have $z_i > \mathsf{pay}_i(\pi)$. Because $\vec{z}_C$ is a lower bound for C, and $z_i > \mathsf{pay}_i(\pi)$, for every $i \in C$, then 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 $\mathsf{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{-C})) \geq z_i > \mathsf{pay}_i(\pi)$, for every $i \in C$. Then, it follows that $(G, \vec{\sigma}) \in$ BEN-DEV, which further implies that $\vec{\sigma}$ cannot be in the core of G—a contradiction to our initial hypothesis.

For the right-to-left direction, suppose that there is π in G such that for every coalition $C \subseteq \mathrm{Ag}$ and lower bound $\vec{z}_C \in \mathbb{Q}^C$ for C, there is $i \in C$ such that $z_i \leq \mathsf{pay}_i(\pi)$. We then simply let $\vec{\sigma}$ be any strategy profile such that $\pi = \pi(\vec{\sigma})$. Now, let $C = \{j, \ldots, k\} \subseteq \mathrm{Ag}$ be any coalition and $\vec{\sigma}'_C$ be any possible deviation of C from $\vec{\sigma}$. Either $\vec{z'}_C = (\mathsf{pay}_j(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C)), \ldots, \mathsf{pay}_k(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C)))$ is a lower bound for C or it is not.

If we have the former, by hypothesis, we know that there is $i \in C$ such that $\mathsf{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C)) \leq \mathsf{pay}_i(\pi)$. Therefore, i will not have an incentive to deviate along with $C \setminus \{i\}$ from $\vec{\sigma}$, and as a consequence coalition C will not be able to beneficially deviate from $\vec{\sigma}$.

If, on the other hand, $\vec{z'}_C$ is not a lower bound for C, then, by the definition of lower bounds, we know that it is not the case that $\vec{\sigma}'_C$ is a joint strategy for C such that for all strategies $\vec{\sigma}'_{-C}$ for $\mathrm{Ag} \setminus C$, we have $\mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C})) \geq \mathsf{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C))$, for every $i \in C$. That is, there exists $i \in C$ and $\vec{\sigma}'_{-C}$ for $\mathrm{Ag} \setminus C$ such that $\mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C})) < \mathsf{pay}_i(\pi(\vec{\sigma}_{-C}, \vec{\sigma}'_C))$. We will now choose $\vec{\sigma}'_{-C}$ so that, in addition, $\mathsf{pay}_i(\pi) \geq \mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{-C}))$ for some i.

Let $\vec{z''}_C = (\mathsf{pay}_j(\pi(\vec{\sigma}^j_{-C}, \vec{\sigma}'_C)), \ldots, \mathsf{pay}_k(\pi(\vec{\sigma}^k_{-C}, \vec{\sigma}'_C)))$ where $\mathsf{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C))$ is defined to be $\min_{\vec{\sigma}'_{-C} \in \Sigma_{-C}} \mathsf{pay}_i(\pi((\vec{\sigma}'_{-C}, \vec{\sigma}'_C)))$. That is, $\vec{\sigma}^i_{-C}$ is a strategy for $\mathrm{Ag} \setminus C$ which ensures the lowest mean-payoff for i assuming that C is playing the joint strategy $\vec{\sigma}'_C$. By construction $\vec{z''}_C$ is a lower bound for C— since each $z''_i = \mathsf{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C))$ is the greatest mean-payoff value that i can ensure for itself when C is playing $\vec{\sigma}'_C$, no matter what coalition $\mathrm{Ag} \setminus C$ does—and therefore, by hypothesis we know that for some $i \in C$ we have $\mathsf{pay}_i(\pi(\vec{\sigma}^i_{-C}, \vec{\sigma}'_C)) \leq \mathsf{pay}_i(\pi)$. As a consequence, as before, i will not have an incentive to deviate along with $C \setminus \{i\}$ from $\vec{\sigma}$, and therefore coalition C will not be able to beneficially deviate from $\vec{\sigma}$. Because C and $\vec{\sigma}'_C$ where arbitrarily chosen, we conclude that $\vec{\sigma} \in \mathrm{CORE}(G)$, proving the right-to-left direction and finishing the proof. □

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 $C \subseteq \mathrm{Ag}$, and vector $\vec{z}_C \in \mathbb{Q}^{\mathrm{Ag}}$.
Question: Is $\vec{z}_C$ is a lower bound for C in G?

Using the MULTI-MEAN-PAYOFF-THRESHOLD decision problem introduced earlier, we can prove the following proposition:

Theorem 6. LOWER-BOUND *is co-NP-complete.*

Proof. First, we show that LOWER-BOUND lies in co-NP by reducing it to MULTI-MEAN-PAYOFF-THRESHOLD. Suppose we have an instance, $(G, C, \vec{z}_C)$, and we want to determine if it is in LOWER-BOUND. We can do this by forming the following two-player, multi-mean-payoff game, $G' = (V_1, V_2, v^0, E, w', z^k)$, where:

- $V_1 = \mathrm{St}$;
- $V_2 = \mathrm{St} \times \mathrm{Ac}_C$;
- $v^0 = s^0$;
- $E = \{(s, (s, \mathrm{ac}_C)) \mid s \in \mathrm{St}, \mathrm{ac}_C \in \mathrm{Ac}_C\}$
  $\cup \{((s, \mathrm{ac}_C), \mathrm{tr}(s, (\mathrm{ac}_C, \mathrm{ac}_{\mathrm{Ag}\setminus C}))) \mid \mathrm{ac}_{\mathrm{Ag}\setminus C} \in \mathrm{Ac}_{\mathrm{Ag}\setminus C}\}$;
- $w' : E \to \mathbb{Z}^{|C|}$, with $w'_i(s, (s, \mathrm{ac}_C)) = w_i(s)$ and,
  $$w'_i((s, \mathrm{ac}_C), \mathrm{tr}(s, (\mathrm{ac}_C, \mathrm{ac}_{\mathrm{Ag}\setminus C}))) = w_i(s);$$
- $z^{|C|} = \vec{z}_c$.

Informally, the two players of the game are C and $\mathrm{Ag} \setminus C$, the vector weight function is given by aggregating the weight functions of C and the threshold is $\vec{z}_C$. Now, if in this game, player 1 has a winning strategy, then there exists some strategy $\vec{\sigma}_C$ such that for all strategies of player 2, $\vec{\sigma}_{\mathrm{Ag}\setminus C}$, we have that $\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})$ is a winning path for player 1. However, this means that $\mathsf{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})) \geq z_i$ for all $i \in C$. However, it is easy to verify that this implies that $\vec{z}_C$ is a lower bound for C in G. Conversely, if player 1 has no winning strategy, then for all strategies, $\vec{\sigma}_C$, there exists some strategy $\vec{\sigma}_{\mathrm{Ag}\setminus C}$ such that $\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})$ is not a winning path. This is turn implies that for some $j \in C$, we have that $\mathsf{pay}_j(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag}\setminus C})) < z_j$, which means that $\vec{z}_C$ is not a lower bound for C in G. Additionally, note that this construction can be performed in polynomial time, giving us the co-NP upper bound. For the lower bound, we go the other way and reduce from MULTI-MEAN-PAYOFF-THRESHOLD.

For the lower bound, we reduce from MULTI-MEAN-PAYOFF-THRESHOLD. Suppose we would like to determine if an instance G is in MULTI-MEAN-PAYOFF-THRESHOLD. Then we form a concurrent mean-payoff game, G', with $k+1$ players, where the states of G' coincide exactly with the states of G. In this game, only the 1st and $(k+1)$th player have any influence on the strategic nature of the game. If the game is in a state in V_1, player one can decide which state to move into next. Otherwise, if the game is in a state within V_2, then the $(k+1)$th player makes a move. Note we only allow moves that agree with moves allowed within G.

Now, in G', the first k players have weight functions corresponding to the k weight functions of player 1 in G. The last player can have any arbitrary weight function. With this machinery in place, we ask if z^k is a lower bound for $\{1, \ldots, k\}$. In a similar manner of reasoning to the above, it is easy to verify that G is an instance of MULTI-MEAN-PAYOFF-THRESHOLD if and only if z^k is a lower bound for $\{1, \ldots, k\}$ in the constructed concurrent mean-payoff game. Moreover, this reduction can be done in polynomial time and we can conclude that LOWER-BOUND is co-NP-complete. □

We have not presented any bounds for the complexity of E-CORE in the general case. One possible reason for the upper bounds remaining elusive to us is due to the fact that whilst in a multi-mean-payoff game, player 2 can act optimally with memoryless strategies, player 1 may require infinite memory [32,33]. Given the close connection between the core in our concurrent, multi-agent setting and winning strategies in multi-mean-payoff games, this raises computational concerns for the E-CORE problem. Additionally, in [35], the authors study the Pareto frontier of multi-mean-payoff games, and provide a way of constructing a representation of it, but this procedure has an exponential time dependency. The same paper also establishes Σ_2^P-completeness for the *polyhedron value problem*. Both of these problems appear to be intimately related to the core, and we hope we might be able to use these results to gain more insight into the E-CORE in the future.

With this having been said, we conclude this section by establishing a link between traditional non-transferable utility (NTU) games and our mean-payoff games—as NTU games are very well studied, and there is a wealth of results relating to core non-emptiness

in this setting [36–38], we hope that some of these results could be utilised in order to understand the core of mean-payoff games.

Formally, an n-person game with NTU is a function, $V : \mathcal{P}(\mathrm{Ag}) \to \mathbb{R}^{|\mathrm{Ag}|}$, such that,

1. For all $C \subseteq \mathrm{Ag}$, $V(C)$ is a non-empty, proper, closed subset of $\mathbb{R}^{|\mathrm{Ag}|}$;
2. For all $C \subseteq \mathrm{Ag}$, if we have $x \in V(C)$ and $y \in \mathbb{R}^{|\mathrm{Ag}|}$ such that $y_i \leq x_i$ for all $i \in C$, then we have $y \in V(C)$;
3. We have that $V(\mathrm{Ag}) \setminus \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\})$ is non-empty and bounded.

We begin by giving a translation from mean-payoff games to NTU games. Let G be a mean-payoff game; then we define an NTU game, $G_{\mathrm{NTU}} = V : \mathcal{P}(\mathrm{Ag}) \to \mathbb{R}^{|\mathrm{Ag}|}$ as follows. If $C \subseteq \mathrm{Ag}$, then,

$$V(C) = \left\{ \vec{z} \in \mathbb{R}^{|\mathrm{Ag}|} \mid \exists \vec{\sigma}_C \forall \vec{\sigma}_{\mathrm{Ag} \setminus C} \forall i \in C, \mathsf{pay}_i\Big(\pi\Big(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag} \setminus C}\Big)\Big) \geq z_i \right\}.$$

In words, $V(C)$ consists of the set of lower bounds that C can force. Note that for an outcome $x \in V(C)$, the components x_i for $i \in \mathrm{Ag} \setminus C$ do not matter—they can be arbitrary real numbers.

Lemma 6. *Let G be a game, and let G_{NTU} be the NTU game associated with G. Then G_{NTU} is well-defined.*

Proof. We need to show that the three conditions in the definition of an NTU game hold for G_{NTU}.

For condition (1), we see that $V(C)$ is always non-empty by noting a coalition can always force an outcome where they achieve at least their worst possible payoff each (the vector made up of each player's lowest weight in the game). The fact that $V(C)$ is closed follows from Theorem 4 of [35]. We also see that $V(C)$ is a proper subset of $\mathbb{R}^{|\mathrm{Ag}|}$, as the members of C can do no better than achieve their maximum weights.

For condition (2), suppose we have $x \in V(C)$, and $y \in \mathbb{R}^{|\mathrm{Ag}|}$ with $y_i \leq x_i$ for all $i \in C$. If $x \in V(C)$, then there exists some $\vec{\sigma}_C$, such that for all $\vec{\sigma}_{\mathrm{Ag} \setminus C}$, we have $\mathsf{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag} \setminus C})) \geq x_i$ for all $i \in C$. However, this in turn implies that $\mathsf{pay}_i(\pi(\vec{\sigma}_C, \vec{\sigma}_{\mathrm{Ag} \setminus C})) \geq y_i$ for all $i \in C$. Thus, by definition, we have $y \in V(C)$.

For condition (3), let p_j be the *punishment value* of the player j in the game G. Informally, the punishment value of a player j can be thought of as the worst payoff that the other players can inflict on that player. Alternatively, we can view the punishment value for player j as the best payoff they can guarantee themselves, no matter what the remaining players do—in this way, we can see that the punishment value is a maximal lower bound for a player.

Consider the vector $p \in \mathbb{R}^{|\mathrm{Ag}|}$, where the jth component of this vector is the punishment value for player j. Naturally, this vector lies in $V(\mathrm{Ag})$. Additionally, we claim that is does not lie in $\operatorname{int} V(\{i\})$ for any $i \in \mathrm{Ag}$. For a contradiction, suppose there existed some $j \in \mathrm{Ag}$ with $p \in \operatorname{int} V(\{j\})$. So there exists some $\epsilon > 0$, such that for all $0 \leq r < \epsilon$, there exists some strategy, σ_j^r, such that for all counterstrategies, $\vec{\sigma}_{-j}$, we have $\mathsf{pay}_j(\pi(\sigma_j^r, \vec{\sigma}_{-j})) \geq p_j + r$. However, this implies player j can achieve a better payoff than their punishment value—a contradiction. Thus, we see that the set $V(\mathrm{Ag}) \setminus \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\})$ is non-empty.

Finally, to see that $V(\mathrm{Ag}) \setminus \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\})$ is bounded, we claim that it is contained in a closed ball of radius M, where M is defined to be:

$$M = \max_{\substack{i \in \mathrm{Ag} \\ s \in \mathrm{St}}} |w_i(s)|.$$

We show that if $x \in V(\mathrm{Ag})$, then we either have $x \in \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\})$ or $x \in \overline{B(0, M)}$, i.e., the closed ball of radius M, centred at the origin.

If $x \in V(\mathrm{Ag})$, then by definition, we must have $x_i \leq M$ for all $i \in \mathrm{Ag}$. Now, there are two possibilities: if we have $x_i \geq -M$ for all $i \in \mathrm{Ag}$, then we have $x \in \overline{B(0,M)}$. So instead suppose there exists some $i \in \mathrm{Ag}$ such that $x_i < -M$. In this case, letting ϵ be any positive number such that $x_i + \epsilon \leq -M$, any strategy σ_i has the property that for all counter-strategies $\vec{\sigma}_{-i}$, we have $\mathsf{pay}_i(\pi(\vec{\sigma}_{-i}, \sigma_i)) \geq x_i + \epsilon$. Thus, we have $x \in \operatorname{int} V(\{i\})$. This implies that,

$$V(\mathrm{Ag}) \subseteq \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\}) \cup \overline{B(0,M)},$$

which in turn implies,

$$V(\mathrm{Ag}) \setminus \bigcup_{i \in \mathrm{Ag}} \operatorname{int} V(\{i\}) \subseteq \overline{B(0,M)},$$

yielding the result. □

Given that we can translate mean-payoff games into well-defined NTU games, it is natural to ask whether we can use traditional cooperative game theory in order to understand the core in our setting. Thus, we introduce the (classic) definition of the core for NTU games. In an NTU game, we say that an element $x \in \mathbb{R}^{|\mathrm{Ag}|}$ is in the core if $x \in V(\mathrm{Ag})$, and there exists no $C \subseteq \mathrm{Ag}$ and no $y \in V(C)$ such that $x_i < y_i$ for all $i \in C$. In the following result, we show that the core of a mean-payoff game, and the core of its corresponding NTU game are intimately related:

Lemma 7. *Let G be a mean-payoff game. Let G_{NTU} be the NTU game associated with G. Then the core of G is non-empty if and only if the core of G_{NTU} is non-empty.*

Proof. First suppose that G has a non-empty core. Thus, there exists some strategy profile $\vec{\sigma}$ such that for all coalitions C and for all strategy vectors $\vec{\sigma}'_C$, there exists some $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$ such that $\mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})) \leq \mathsf{pay}_i(\pi(\vec{\sigma}))$ for some $i \in C$. Let $x \in \mathbb{R}^{\mathrm{Ag}}$ be such that $x_i = \mathsf{pay}_i(\vec{\sigma})$ for all $i \in \mathrm{Ag}$. Then by definition, we have $x \in V(\mathrm{Ag})$. We claim that x is in the core of G_{NTU}. Suppose there is some $C \subseteq \mathrm{Ag}$ and a $y \in V(C)$ such that $x_i < y_i$ for all $i \in C$. Thus, there exists some $\vec{\sigma}'_C$ such that for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, such that $\mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})) \geq y_i > x_i = \mathsf{pay}_i(\vec{\sigma})$ for all $i \in C$. However, this implies that $\vec{\sigma}$ is not in the core of G, which is a contradiction. Thus, x is in the core of G_{NTU}.

Conversely, suppose that G_{NTU} has an empty core. Thus, there exists some $x \in \mathbb{R}^{|\mathrm{Ag}|}$ such that $x \in V(\mathrm{Ag})$, such that there exists no $C \subset \mathrm{Ag}$ and no $y \in V(C)$ with $x_i < y_i$ for all $i \in C$. Since $x \in V(\mathrm{Ag})$, there exists some strategy $\vec{\sigma}$ such that $\mathsf{pay}_i(\pi(\vec{\sigma})) \geq x_i$ for all $i \in \mathrm{Ag}$. We claim that $\vec{\sigma}$ is in the core of G. If it were not, then there would exist some coalition C, and some strategy vector $\vec{\sigma}'_C$ such that for all strategy vectors $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, we have $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}) \succ_i \pi(\vec{\sigma})$ for all $i \in C$. We then define $y \in \mathbb{R}^{\mathrm{Ag}}$ by setting $y_i = \min_{\vec{\sigma}'_{\mathrm{Ag}\setminus C}} \mathsf{pay}_i(\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C}))$ for $i \in C$ and setting $y_i = 0$ for $i \in \mathrm{Ag} \setminus C$. Then we have that $y \in V(C)$ by definition. Since for all $\vec{\sigma}'_{\mathrm{Ag}\setminus C}$, we have that $\pi(\vec{\sigma}'_C, \vec{\sigma}'_{\mathrm{Ag}\setminus C})$ is strictly preferred to $\pi(\vec{\sigma})$ by all players in C, we must have that $y_i > x_i$ for all $i \in C$. However, this contradicts the fact that x is in the core of G_{NTU}. Thus, we must have that $\vec{\sigma}$ is in the core of G. □

As stated previously, we have been unable to determine the complexity of E-CORE in the setting of mean-payoff games. However, given the above result, we suggest a route which may bear fruits in the future. In [36–40], the authors reason about the core of cooperative games (in both the transferable utility and non-transferable utility settings) by appealing to the notion of a *balanced* set. In [37], the authors generalise this by introducing the notion of π-balancedness. Let $\pi = \{\pi_C \in \mathbb{R}^{|\mathrm{Ag}|} \mid C \subseteq \mathrm{Ag}\}$ be a collection of vectors such that:

1. For all $C \subseteq \mathrm{Ag}$, we have $\pi_C \neq \mathbf{0}$;
2. For all $C \subseteq \mathrm{Ag}$, and for all $i \notin C$, we have $\pi_{C,i} = 0$;

3. For all $C \subseteq \mathrm{Ag}$, and for all $i \in C$, we have $\pi_{C,i} \geq 0$,

and let $\mathcal{C} \subseteq \mathcal{P}(\mathrm{Ag})$ be a collection of coalitions. We say that $\mathcal{C}$ is π-balanced if there exist balancing weights, $\lambda_C > 0$, for each $C \subseteq \mathrm{Ag}$ such that:

$$\sum_{C \in \mathcal{C}} \lambda_C \pi_C = \pi_{\mathrm{Ag}}.$$

We then say that an NTU game, V, is π-balanced if whenever $\mathcal{C}$ is a π-balanced collection, we have:

$$\bigcap_{C \in \mathcal{C}} V(C) \subseteq V(\mathrm{Ag}).$$

In [37], the authors show that if there exists some π such that V is π-balanced, then V has a non-empty core. The condition of π-balancedness translates readily over to the setting of mean-payoff games, and so we see that if such a game is π-balanced, then it has a non-empty core. This suggest a (sound, but not complete) algorithm for detecting if a mean-payoff game has a non-empty core; somehow guess a polynomial-sized π, use a linear program to calculate the corresponding balancing weights, and then use an co-NP oracle to verify there exists no π-balanced collection such that $\bigcap_{C \in \mathcal{C}} V(C) \subseteq V(\mathrm{Ag})$. Obviously, this is not a rigorous argument, but is suggestive of what a possible solution may look like.

Additionally, whilst π-balancedness is a sufficient condition for core non-emptiness, it is not necessary. However, in [37], the authors strengthen the condition of π-balancedness in the setting of convex-valued NTU games, to obtain a necessary and sufficient result. Given in that mean-payoff games, the outcomes that a coalition can achieve can be expressed as a union of convex sets, this approach seems promising. However, we have been unable to yield any results via this route.

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, $\mathrm{tr} : \mathrm{St} \times \mathrm{Ac}_1 \times \cdots \times \mathrm{Ac}_{|\mathrm{Ag}|} \to \mathrm{St}$ is a total function, so it has size $|\mathrm{Ac}|^{|\mathrm{Ag}|}$. Thus, the size of the game scales exponentially with the number of the agents. In Example 1, the underlying concurrent game structure has a size of 429,981,696. If we are ever to have computational tools to support the decision problems described in this paper, then such "extensive" representations are not viable: we will require compact frameworks to represent games.

One natural framework we can use to induce succinctness is that of *Reactive Modules* [41]. Specifically, we modify the Reactive Module Games of [18] with weights on the guarded commands. We begin by walking through some preliminaries.

Reactive modules 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 $\mathrm{guard}(g)$, and we call the variables (the x_js) on the right-hand-side of g the *controlled variables* of g, denoted by $\mathrm{ctr}(g)$. 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 $\mathrm{guard}(g)$ is true (we say that they are enabled for execution). Each module can then choose one enabled command, g, and reassign the variables in $\mathrm{ctr}(g)$ 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.

To define the size of a guarded command, we first define the size of a propositional formula, φ, denoted $|\varphi|$, to be the number of logical connectives it contains. Then the size of a guarded command g, written $|g|$, is given by $|\text{guard}(g)| + |\text{ctr}(g)|$.

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 $\text{ctr}(g) \subseteq \Psi$.

After defining a simple reactive module m, we introduce an additional command, $g^{\textbf{skip}}$, of the form,

$$\bigwedge_{g \in U} \neg \text{guard}(g) \rightsquigarrow \emptyset$$

with $\text{ctr}(g) = \Psi$. The empty set on the right hand side of the guarded command means that no variables are changed. We introduce this extra guarded command so that at each stage, every module has *some* action they can take. However, this is not obligatory, and if we define a reactive module where we can prove at each step, there will be an available action, then introducing $g^{\textbf{skip}}$ is not necessary. With this added, the *size* of a reactive module, $|m|$, is given by the sum of the sizes of its constituent guarded commands.

Given this, 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. We define the *size* of an arena to be sum of the sizes of the modules within it.

With this syntactic machinery in place, we are finally ready to describe the semantics of SRML arenas. We give a brief, high-level description here—for details, please refer to [18,42].

With this syntactic machinery in place, we are finally ready to describe the semantics of SRML arenas. Let $v \subseteq \Phi$ be a valuation and let $i \in \text{Ag}$ be an agent. We let $\text{enabled}_i(v)$ denote the guarded commands available to agent i under the valuation v. Formally, we have:

$$\text{enabled}_i(v) = \{g \in U_i \mid vs. \models \text{guard}(g)\}.$$

We then define $\text{enabled}(v) = \text{enabled}_1(v) \times \cdots \times \text{enabled}_{|\text{Ag}|}(v)$. Given that each U_i contains a command of the form $g_i^{\textbf{skip}}$ as described previously, we can see that each $\text{enabled}_i(v)$ is always non-empty.

We also define $\text{exec}_i(g, v)$, which given a valuation v, is the valuation of Φ_i given by executing $g = \varphi \rightsquigarrow x_1' := \psi_1; \ldots; x_k' := \psi_k$. Specifically, we have:

$$\text{exec}_i(g, v) = (v_i \setminus \text{ctr}(g)) \cup \{x_j \mid vs. \models \psi_j\}.$$

Thus, upon executing g, agent i's variables are reassigned according to the valuations of the propositional formulae on its right hand side. However, with multiple agents, there is some strategic interaction, and we wish to understand how the actions of all the agents affect the state of the system. As such, we define *joint guarded commands*, which are simply a selection of guarded commands, one for each agent. That is, we write $J = g_1 \times \cdots \times g_{|\text{Ag}|}$. Similarly to before, we set $\text{exec}(J, v) = \text{exec}_1(g_1, v) \cup \ldots \cup \text{exec}_{|\text{Ag}|}(g_{|\text{Ag}|}, v)$.

We are now ready to describe how arenas 'play out'. Initially, each agent i picks a guarded command $g_i^0 \in I_i$ and this sets each of their variables accordingly, inducing a valuation $v^0 = \text{exec}(J^0, \Phi)$. The agents then each pick a guarded command $g_i^1 \in \text{enabled}_i(v^0)$ and execute them, inducing another valuation, $v^1 = \text{exec}(J^1, v^0)$. They repeat this ad infinitum. As such, this induces a *path* of the game. However, unlike before, where we

defined paths over states of the game, here, we define paths over joint guarded commands, $\pi : \mathbb{N} \to (I_1 \cup U_1) \times \cdots \times (I_{|\mathrm{Ag}|} \cup U_{|\mathrm{Ag}|})$. 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 know infer 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 the 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), $G = (A, \{w_i\}_{i \in \mathrm{Ag}})$, is an SRML arena, $A = (\mathrm{Ag}, \Phi, \{m_i\}_{i \in \mathrm{Ag}})$, 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 path.

Note that we are effectively assigning weights to transitions, rather than to states as we did before. This is not a huge conceptual shift, and moving between the two representations (weights on states and weights on transitions) is a relatively straightforward transformation.

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 $\mathrm{Inf}(p)$ with $p \in \Phi$. Moreover, since if p is a proposition, then $\neg p$ is also a proposition, we use the shorthand $\mathrm{Inf}(\neg p)$, with the obvious interpretation, in place of the previous $\mathrm{Inf}(\bar{F})$ notation. The semantics of these specifications are defined in a nearly identical way to before.

Before considering the decision problems relating to WRMGs, let us now walk through an example to demonstrate their conciseness and utility of WRMGs. We do this by revisiting Example 1.

Example 2. *In Example 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 $\Phi_i = \{x_{i,1}, \ldots, x_{i,12}, p_i, d_i, c_i\}$, 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:*

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

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.

We can also rewrite the specification of Equation (1) in our new setting as follows:

$$\bigwedge_{i\in[4]} Inf(p_i) \wedge Inf(\neg p_i).$$

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. Moreover, our specification has not increased in size.*

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 $\pi(\vec{\sigma}) \models \alpha$?

Theorem 7. *The* WRMG-E-NASH *problem lies in NEXPTIME and is EXPTIME-hard.*

Proof. The idea is to 'blow up' the simple reactive module arena into a concurrent game structure, then apply the same techniques as before. Explicitly, given a WRMG, $G = (\mathrm{Ag}, \Phi, \{m_i\}_{i\in\mathrm{Ag}}, \{w_i\}_{i\in\mathrm{Ag}})$, we form a graph $\mathcal{G} = (V, E)$ as follows. We set $V = \mathcal{P}(\Phi \cup \{p_{\mathrm{init}}\})$ and then introduce an edge, $e = (v, w)$ if there exists some joint guarded update command J such that $\mathrm{exec}(J, v) = w$. We also introduce edges $(\{p_{\mathrm{init}}\}, v)$, if there exists some joint guarded initialisation command J such that $\mathrm{exec}(J, \emptyset) = v$. We then form a concurrent game structure, $G' = (\mathrm{Ag}', \mathrm{St}, s^0, (\mathrm{Ac}_i)_{i\in\mathrm{Ag}}, \mathrm{tr})$ in the natural way. We set $\mathrm{Ag}' = \mathrm{Ag}$ and $\mathrm{St} = V$. We also set $s^0 = \{p_{\mathrm{init}}\}$. The actions for each player, Ac_i, consist of their respective guarded commands and the transition function corresponds to the edges of the graph. The weights of the guarded commands are attached to the weights of the transitions in the concurrent game structure (Whilst our analysis of E-NASH is couched in terms of weights on the states, in the proof of it, we modify the concurrent game structure so the weights exist on the transitions instead. Thus, this reduction does not need to introduce extra states to convert the weights on the transitions into weights on the states—we simply 'cut the corner' instead).

We also need to transform the ω-regular specification on the weighted reactive module games, α, to a specification on the concurrent game structure, α'. To do this, for each propositional variable p, define a subset $F_p \subseteq \mathrm{St}$ where $v \in F_p$ if $p \in v$. Then we simply replace every occurrence of $\mathrm{Inf}(p)$ in α with $\mathrm{Inf}(F_p)$.

It should be apparent that (G', α') exhibits the same behaviours and qualities as (G, α) and is exponential in size relative to (G, α). With this expansion complete, we can straightforwardly apply the NP algorithm of Theorem 2, immediately giving us the NEXPTIME upper bound.

To show hardness, we need to introduce another decision problem, PEEK-G4 [43]. A PEEK-G4 instance is a tuple (X_1, X_2, X_3, φ). Here, X_1 and X_2 are finite, disjoint, non-empty sets of propositional variables, $X_3 \subseteq X_1 \cup X_2$ gives the variables that are true in the initial state, and φ is a propositional formula over $X_1 \cup X_2$. The game is played by starting in the configuration given by X_3. The players then alternate, either choosing to toggle a variable they control, or skipping a go. If on their turn, a player can make φ true, then they win and the game is over. The decision problem associated with this game is to determine if agent 2 has a winning strategy for a give tuple (X_1, X_2, X_3, φ).

We reduce an instance of PEEK-G4, (X_1, X_2, X_3, φ), to an instance of WRMG-E-NASH, $(\mathrm{Ag}, \Phi, \{m_i\}_{i\in\mathrm{Ag}}, \{w_i\}_{i\in\mathrm{Ag}})$. We set

$$\mathrm{Ag} = \{0, 1, 2\} \text{ and } \Phi = (X_1 \cup X_2) \cup \{\mathsf{turn_p}, \mathsf{turn_u}, \mathsf{win}_1, \mathsf{win}_2\}$$

and introduce three reactive modules over Φ, m_0, m_1 and m_2. The modules m_1 and m_2 represent the two players, 1 and 2, whilst m_0 represents an umpire/officiator, who keeps track of whose turn it is and whether anyone has won the game yet.

The two players alternate, with the umpire enforcing this. That is, for m_0, we have $\Phi_0 = \{\mathsf{turn}_\mathrm{p}, \mathsf{turn}_\mathrm{u}, \mathsf{win}_1, \mathsf{win}_2\}$, with the following single initialisation command:

$$\top \rightsquigarrow \mathsf{turn}'_\mathrm{p} := \top; \mathsf{turn}'_\mathrm{u} := \bot; \mathsf{win}'_1 := \bot; \mathsf{win}'_2 := \bot.$$

The turn_p variable keeps track of which player's turn it is, and the turn_u variable determined if it is the umpire's turn. Finally, the two win variables keep track of whether anyone has won yet. The update commands for m_0 are defined as follows:

$$\begin{aligned}
\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \wedge \neg\mathsf{turn}_\mathrm{u} &\rightsquigarrow \mathsf{turn}'_\mathrm{u} := \top,\\
\mathsf{turn}_\mathrm{u} \wedge \neg\varphi &\rightsquigarrow \mathsf{turn}'_\mathrm{u} := \bot; \mathsf{turn}'_\mathrm{p} := \neg\mathsf{turn}_\mathrm{p},\\
\mathsf{turn}_\mathrm{u} \wedge \varphi \wedge \mathsf{turn}_\mathrm{p} &\rightsquigarrow \mathsf{win}'_1 := \top,\\
\mathsf{turn}_\mathrm{u} \wedge \varphi \wedge \neg\mathsf{turn}_\mathrm{p} &\rightsquigarrow \mathsf{win}'_2 := \top,\\
\mathsf{win}_1 \vee \mathsf{win}_2 &\rightsquigarrow \emptyset.
\end{aligned}$$

There is a lot going on in the above set of commands, so let us break it down: the first two commands say that play alternates between the players and the umpire—one of the players makes a move, then play goes to the umpire, who checks for wins, then play passes to the other player, and so on. The third and fourth commands say that if a player managed to satisfy φ on their last turn, then they have won, and the umpire can declare them the winner. The final command says that the umpire will just stay put and keep their variables constant once one player has won.

The two player modules are symmetric, so we define m_1, with m_2 being formed in a similar way. For m_1, we have $\Phi_1 = X_1 \cup \{\mathsf{win}_2\}$, with a single initialisation command:

$$\top \rightsquigarrow x'_{i_1} := \top; \ldots; x'_{i_n} := \top,$$

where $\{x_{i_k}\}_{1 \leq k \leq n} = X_1 \cap X_3$. This simply says that player 1 must set their variables as dictated by X_3 Then for every variable $x_i \in X_1$, we introduce an update guarded command of the following form,

$$\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \wedge \mathsf{turn}_\mathrm{p} \wedge \neg\mathsf{turn}_\mathrm{u} \rightsquigarrow x'_i := \neg x_i.$$

This says that if neither player has won yet, and it is player 1's turn to play, then they can toggle one of the variables they control. Additionally, we need to allow a player to be able not to change any of their variables, and we do this with the following guarded command,

$$\neg(\mathsf{win}_1 \vee \mathsf{win}_2) \rightsquigarrow \emptyset.$$

Note that player 1 can always play this command, even when it is not their turn. Finally, we need to introduce two more guarded commands that the players take when the game is over:

$$\begin{aligned}
\mathsf{win}_1 &\rightsquigarrow \emptyset\\
\mathsf{win}_2 &\rightsquigarrow \emptyset.
\end{aligned}$$

If someone has won the game, then both players ignore the turn system and have the lone option of following exactly one of the above two commands forever.

With this defined, we need to introduce weight functions for each of the three players. For the umpire, we give a constant weight function (so they are ambivalent between all outcomes) and for player 1 we give them a weight function w_1 such that,

$$\begin{aligned} w_1(\mathsf{win}_1 \rightsquigarrow \emptyset) &= 1, \\ w_1(\mathsf{win}_2 \rightsquigarrow \emptyset) &= -1, \\ w_1(g) &= 0, \text{ for all other } g. \end{aligned}$$

with player 2's weight function being defined in the dual way. Finally, the ω-regular specification is simply $\alpha = \mathrm{Inf}(\mathsf{win}_2)$. We claim that player 2 has a winning strategy in PEEK-G4 if and only if there exists a Nash equilibrium in the weighted reactive module game that models the given specification. Moreover, it is patently clear that this construction can be performed in polynomial time.

First suppose that player 2 has a winning strategy in PEEK-G4. Then m_2 can play this strategy and m_1 can play any arbitrary strategy and eventually the game will end up in state win_2. From here, each player only has one guarded command they can play, meaning player 2 will get a payoff of 1 and player 1 will get a payoff of -1. This is a Nash equilibrium. No matter what m_1 plays, they are unable to force a win against player 2's winning strategy and so have no incentive to deviate. Additionally, player 2 is achieving their maximum payoff, and thus have no motivation to deviate. This strategy profile also models α.

Conversely, suppose there exists some Nash equilibrium which models α. This implies that the game eventually ends up in win_2, and so player 2 has a payoff of 1 and player 1 a payoff of -1 in this equilibrium. The fact that this is an equilibrium implies that no matter what player 1 plays, they cannot increase their payoff, i.e. they always lose against the strategy player 2 is playing. This implies that player 2 has a winning strategy in PEEK-G4 and this concludes the proof. □

6. Related Work

6.1. Contributions of This Paper

This paper introduces ω-regular specifications as a natural, expressive, computationally tractable way of reasoning qualitatively about concurrent games. In particular, we establish several results within the rational verification framework, the most important of which are the complexity bounds for the E-NASH, E-CORE and WRMG-E-NASH problems.

6.2. Mean-Payoff Games

Mean-payoff games are a useful tool in the analysis 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 [6].

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 [5], and two-player multi-mean-payoff games [32], where the focus is on the computation of winning strategies for either player, a problem that can be solved in co-NP in the general case, and in NP for memoryless strategies.

6.3. Combined Qualitative and Quantitative Reasoning

Combining qualitative and quantitative reasoning has mainly been done by modifying players' mean-payoff with some qualitative measure. In [44], the authors consider two-player, zero-sum games, where on each path 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, [30,31] 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, [15] 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.

6.4. ω-Regular Specifications

Games with ω-regular objectives have been studied mostly in the context of two-player games [8], 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 rgames 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.

6.5. On Rational Verification

The problem studied in this paper is called Rational Verification, which has been studied for different types of games and specification languages [10]. While rational verification with LTL goals is 2EXPTIME-complete, the problem can become considerably easier when considering simpler specification languages [15]. However, in the context of multi-player mean-payoff games, only a solution for generalised Büchi specification 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 [17], where the core for concurrent game structures was introduced. Furthermore, regarding the latter, a comprehensive study using reactive modules games can be found in [18].

6.6. Future Work

There are multiple interesting avenues for future research. The two most obvious open problems are that of determining the complexity bounds for the E-CORE problem in the general case, as well as closing the complexity gap for the WRMG-E-NASH problem. Following this, there are other directions that appear to be fruitful, as well as interesting and worthwhile—for example, introducing both imperfect information and nondeterminism offers a closer approximation to real-world systems, as well as raising interesting mathematical questions. We are also interested in using ω-regular specifications to understand ω-regular games in a unified, principled way.

Author Contributions: Conceptualisation, J.G., T.S. and M.W.; writing, J.G., T.S. and M.W. All authors have read and agreed to the published version of the manuscript.

Funding: 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.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

Glossary

P	The class of decision problems which can be solved in polynomial time by a deterministic Turing machine
NP	The class of decision problems which can be solved in polynomial time by a non-deterministic Turing machine
co-NP	The complement of the complexity class NP
PSPACE	The class of decision problems which can be solved using polynomial space by a deterministic Turing machine
NEXPTIME	The class of decision problems which can be solved in exponential time by a non-deterministic Turing machine
2EXPTIME	The class of decision problems which can be solved in doubly exponential time by a deterministic Turing machine
TFNP	The class of total function problems which can be solved in polynomial time by a non-deterministic Turing machine
Nash equilibrium	Informally, a strategy profile which is resistant to unilateral deviations
The core	Informally, the set of strategy profiles which are resistant to multilateral deviations, assuming the remaining players can respond to such a deviation
Σ_2^P	The class of decision problems which can be solved in polynomial time by a non-deterministic Turing machine with an NP oracle
ω-regular specifications	A formalism which captures infinitely repeating behaviours—they look like Boolean combinations of atoms of the form $\mathrm{Inf}(F)$, where F is some subset of the states of the game; their syntax and semantics are defined in Section 2

Acronyms

LTL	Linear temporal logic
CTL	Computation tree logic
GR(1)	Generalised reactivity(1)
NTU	Non-transferable utility
SRML	Simple reactive modules lanuage
WRMG	Weighted reactive module game

Notation

$\lvert S \rvert$	The size of the set S
σ_i	A strategy for a player i
$\vec{\sigma}_C$	A strategy vector for a coalition C
$\vec{\sigma}$	A strategy profile—i.e., a strategy for each player
π	A path—a finite or infinite sequence of states; also used in the context of NTU games in the traditional definition of 'balancedness'
$\pi(\vec{\sigma})$	The path induced by the players following the strategy profile $\vec{\sigma}$
α	An acceptance condition
G	A game—the type of which depends on the context
$\emptyset$	The empty set
$\top$	The logical formula, true
$\bot$	The logical formula, false
φ	A logical formula—the type of which depends on the context
$\mathrm{int}(S)$	The interior of the set S
pun_i	The largest payoff player i can achieve, no matter what the remaining players do
Φ	The set of all propositional variables under consideration in a given context
$\mathrm{Inf}(F)$	An atom in an ω-regular expression; informally, interpreted to mean 'the set F is visited infinitely often'

$\mathrm{Fin}(F)$	An atom in an ω-regular expression; informally, interpreted to mean 'the set F is only visited finitely often'
$\mathbb{Z}$	The integers: $\ldots, -1, 0, 1, \ldots$
$\mathbb{N}$	The natural numbers: $0, 1, \ldots$
$\mathsf{mp}(\beta)$	The mean-payoff of an infinite sequence of real numbers, β; formally defined in Section 2, but can be thought of informally as a sort of infinite average
$\mathrm{NE}(G)$	The set of Nash equilibria of a game G
$\models$	The modelling relation—informally, $\pi \models \alpha$ means that the path π adheres to the behaviour described in α; defined formally in Section 2
w_i	The weight function of player i, which defines that player's preferences
M	A concurrent game structure
Ag	The set of agents in a given concurrent game structure
St	The set of game states in a given concurrent game structure
s^0	The start state of a concurrent game structure
Ac_i	The actions available to player i in a given concurrent game structure
tr	The transition function of a given concurrent game structure—takes the current state of the game, along with an action for each agent/player, and outputs a new state for the game
A	An SRML arena
m_i	A reactive module corresponding to player i
U_i	A set of 'initialisation commands' in the reactive module corresponding to player i
U_i	A set of 'update commands' in the reactive module corresponding to player i
$\mathrm{guard}(g)$	The 'guard' of the guarded command g—that is, the precondition for taking the action defined by g
$\mathrm{ctr}(g)$	The 'controlled' of the guarded command g—that is, the propositional variables that will be affected upon executing g
$\mathrm{enabled}_i(v)$	The guarded commands available to player i under the valuation v
$\mathrm{exec}_i(g, v)$	The valuation of player i's variables obtained by executing the guarded command g under the valuation v

References

1. Alur, R.; Henzinger, T.A.; Kupferman, O. Alternating-time temporal logic. *J. ACM* **2002**, *49*, 672–713. [CrossRef]
2. Alfaro, L.d.; Henzinger, T.A. Concurrent Omega-Regular Games. In Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science, Santa Barbara, CA, USA, 26–29 June 2000; IEEE Computer Society: Washington, DC, USA, 2000; pp. 141–154. [CrossRef]
3. Henzinger, T.A. Games in system design and verification. In Proceedings of the 10th Conference on Theoretical Aspects of Rationality and Knowledge (TARK-2005), Singapore, 10–12 June 2005; Meyden, R.v.d., Ed.; National University of Singapore: Singapore, 2005; pp. 1–4.
4. Ehrenfeucht, A.; Mycielski, J. Positional strategies for mean payoff games. *Int. J. Game Theory* **1979**, *8*, 109–113. [CrossRef]
5. Ummels, M.; Wojtczak, D. The Complexity of Nash Equilibria in Limit-Average Games. *arXiv* **2011**, arXiv:1109.6220.
6. Zwick, U.; Paterson, M. The complexity of mean payoff games on graphs. *Theor. Comput. Sci.* **1996**, *158*, 343–359. [CrossRef]
7. Osborne, M.J.; Rubinstein, A. *A Course in Game Theory*; MIT Press: Cambridge, MA, USA, 1994.
8. Chatterjee, K.; Henzinger, T.A. A survey of stochastic ω-regular games. *J. Comput. Syst. Sci.* **2012**, *78*, 394–413. [CrossRef]
9. Fisman, D.; Kupferman, O.; Lustig, Y. Rational Synthesis. *arXiv* **2009**, arXiv:0907.3019.
10. Wooldridge, M.J.; Gutierrez, J.; Harrenstein, P.; Marchioni, E.; Perelli, G.; Toumi, A. Rational Verification: From Model Checking to Equilibrium Checking. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Schuurmans, D., Wellman, M.P., Eds.; AAAI Press: Palo Alto, CA, USA 2016; pp. 4184–4191.
11. Pnueli, A. The Temporal Logic of Programs. In Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Providence, RI, USA, 31 October–1 November 1977; IEEE Computer Society: Washington, DC, USA, 1977; pp. 46–57. [CrossRef]
12. Clarke, E.M.; Emerson, E.A. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Logics of Programs*; Springer: Berlin/Heidelberg, Germany, 1981; pp. 52–71. [CrossRef]
13. Pnueli, A.; Rosner, R. On the Synthesis of an Asynchronous Reactive Module. In *Automata, Languages and Programming, Proceedings of the 16th International Colloquium, ICALP89, Stresa, Italy, 11–15 July 1989*; Ausiello, G., Dezani-Ciancaglini, M., Rocca, S.R.D., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1989; Volume 372, pp. 652–671. [CrossRef]

14. Bloem, R.; Jobstmann, B.; Piterman, N.; Pnueli, A.; Sa'ar, Y. Synthesis of Reactive(1) designs. *J. Comput. Syst. Sci.* **2012**, *78*, 911–938. [CrossRef]
15. Gutierrez, J.; Najib, M.; Perelli, G.; Wooldridge, M.J. On Computational Tractability for Rational Verification. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019; Kraus, S., Ed.; pp. 329–335. [CrossRef]
16. Babiak, T.; Blahoudek, F.; Duret-Lutz, A.; Klein, J.; Kretínský, J.; Müller, D.; Parker, D.; Strejcek, J. The Hanoi Omega-Automata Format. In Proceedings of the Computer Aided Verification—27th International Conference, CAV 2015, San Francisco, CA, USA, 18–24 July 2015; Kroening, D., Pasareanu, C.S., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9206, pp. 479–486. [CrossRef]
17. Gutierrez, J.; Kraus, S.; Wooldridge, M.J. Cooperative Concurrent Games. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, 13–17 May 2019; Elkind, E., Veloso, M., Agmon, N., Taylor, M.E., Eds.; pp. 1198–1206. Available online: http://www.cs.ox.ac.uk/people/michael.wooldridge/pubs/aamas2019.pdf (accessed on 21 November 2021).
18. Gutierrez, J.; Harrenstein, P.; Wooldridge, M.J. From model checking to equilibrium checking: Reactive modules for rational verification. *Artif. Intell.* **2017**, *248*, 123–157. [CrossRef]
19. Megiddo, N.; Papadimitriou, C.H. On total functions, existence theorems and computational complexity. *Theor. Comput. Sci.* **1991**, *81*, 317–324. [CrossRef]
20. Nash, J.F. Equilibrium points in n-person games. *Proc. Natl. Acad. Sci. USA* **1950**, *36*, 48–49. [CrossRef]
21. Nash, J. Non-cooperative games. *Ann. Math.* **1951**, *54*, 286–295. [CrossRef]
22. Gillies, D.B. 3. Solutions to General Non-Zero-Sum Games. In *Contributions to the Theory of Games (AM-40), Volume IV*; Tucker, A.W., Luce, R.D., Eds.; Princeton University Press: Princeton, NJ, USA, 1959; pp. 47–86. [CrossRef]
23. Aumann, R.J. Acceptable points in general cooperative *n*-person games. In *Contributions to the Theory of Games, Volume IV*; Annals of Mathematics Studies, No. 40; Princeton University Press: Princeton, NJ, USA, 1959; pp. 287–324.
24. Aumann, R.J. Acceptable points in games of perfect information. *Pac. J. Math.* **1960**, *10*, 381–417. [CrossRef]
25. Allen Emerson, E.; Lei, C.L. Modalities for model checking: Branching time logic strikes back. *Sci. Comput. Program.* **1987**, *8*, 275–306. [CrossRef]
26. Gutierrez, J.; Harrenstein, P.; Wooldridge, M.J. Iterated Boolean games. *Inf. Comput.* **2015**, *242*, 53–79. [CrossRef]
27. Karp, R.M. A characterization of the minimum cycle mean in a digraph. *Discret. Math.* **1978**, *23*, 309–311. [CrossRef]
28. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman: New York, NY, USA, 1979.
29. Karp, R.M. Reducibility among Combinatorial Problems. In Proceedings of the a Symposium on the Complexity of Computer Computations, Yorktown Heights, NY, USA, 20–22 March 1972; Miller, R.E., Thatcher, J.W., Eds.; The IBM Research Symposia Series; Plenum Press: New York, NY, USA, 1972; pp. 85–103. [CrossRef]
30. Gutierrez, J.; Murano, A.; Perelli, G.; Rubin, S.; Wooldridge, M. Nash Equilibria in Concurrent Games with Lexicographic Preferences. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, Melbourne, Australia, 19–25 August 2017; pp. 1067–1073. [CrossRef]
31. Gutierrez, J.; Murano, A.; Perelli, G.; Rubin, S.; Steeples, T.; Wooldridge, M. Equilibria for games with combined qualitative and quantitative objectives. *Acta Inform.* **2020**, *58*, 585–610. [CrossRef]
32. Velner, Y.; Chatterjee, K.; Doyen, L.; Henzinger, T.A.; Rabinovich, A.M.; Raskin, J.F. The complexity of multi-mean-payoff and multi-energy games. *Inf. Comput.* **2015**, *241*, 177–196. [CrossRef]
33. Kopczynski, E. Half-Positional Determinacy of Infinite Games. In *Automata, Languages and Programming, Proceedings of the 33rd International Colloquium, ICALP 2006, Venice, Italy, 10–14 July 2006*; Bugliesi, M., Preneel, B., Sassone, V., Wegener, I., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4052, pp. 336–347. [CrossRef]
34. Sistla, A.P.; Clarke, E.M. The Complexity of Propositional Linear Temporal Logics. *J. ACM* **1985**, *32*, 733–749. [CrossRef]
35. Brenguier, R.; Raskin, J. Pareto Curves of Multidimensional Mean-Payoff Games. In *Computer Aided Verification, Proceedings of the 27th International Conference, CAV 2015, San Francisco, CA, USA, 18–24 July 2015*; Kroening, D., Pasareanu, C.S., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switzerland, 2015; Volume 9207, pp. 251–267. [CrossRef]
36. Scarf, H.E. The core of an N person game. *Econom. J. Econom. Soc.* **1967**, *35*, 50–69. [CrossRef]
37. Billera, L.J. Some theorems on the core of an n-game without side-payments. *SIAM J. Appl. Math.* **1970**, *18*, 567–579. [CrossRef]
38. Shapley, L.S. On Balanced Games without Side Payments. In *Mathematical Programming*; Elsevier: Amsterdam, The Netherlands, 1973; pp. 261–290. [CrossRef]
39. Bondareva, O.N. Some applications of the methods of linear programming to the theory of cooperative games. *Probl. Kibernet.* **1963**, *10*, 119–139.
40. Shapley, L.S. On balanced sets and cores. *Nav. Res. Logist. Q.* **1967**, *14*, 453–460. [CrossRef]
41. Alur, R.; Henzinger, T.A. Reactive Modules. In Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, NJ, USA, 27–30 July 1996; IEEE Computer Society: Washington, DC, USA, 1996; pp. 207–218. [CrossRef]
42. van der Hoek, W.; Lomuscio, A.; Wooldridge, M.J. On the complexity of practical ATL model checking. In Proceedings of the 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006), Hakodate, Japan, 8–12 May 2006; Nakashima, H., Wellman, M.P., Weiss, G., Stone, P., Eds.; pp. 201–208. [CrossRef]

43. Stockmeyer, L.J.; Chandra, A.K. Provably Difficult Combinatorial Games. *SIAM J. Comput.* **1979**, *8*, 151–174. [CrossRef]
44. Chatterjee, K.; Henzinger, T.A.; Jurdzinski, M. Mean-payoff parity games. In Proceedings of the 20th Annual IEEE Symposium on Logic in Computer Science (LICS'05), Chicago, IL, USA, 26–29 June 2005; pp. 178–187.