Annals of Mathematics and Artificial Intelligence (2023) 91:409–430
https://doi.org/10.1007/s10472-022-09804-3

On the complexity of rational verification

Julian Gutierrez[1] · **Muhammad Najib**[2] · **Giuseppe Perelli**[3] · **Michael Wooldridge**[4]

Accepted: 23 June 2022 / Published online: 14 July 2022

Abstract
Rational verification refers to the problem of checking which temporal logic properties hold of a concurrent/multiagent system, under the assumption that agents in the system choose strategies that form a game theoretic equilibrium. Rational verification can be understood as a counterpart to model checking for multiagent systems, but while classical model checking can be done in polynomial time for some temporal logic specification languages such as CTL, and polynomial space with LTL specifications, rational verification is much harder: the key decision problems for rational verification are 2EXPTIME-complete with LTL specifications, even when using explicit-state system representations. Against this background, our contributions in this paper are threefold. First, we show that the complexity of rational verification can be greatly reduced by restricting specifications to GR(1), a fragment of LTL that can represent a broad and practically useful class of response properties of reactive systems. In particular, we show that for a number of relevant settings, rational verification can be done in polynomial space and even in polynomial time. Second, we provide improved complexity results for rational verification when considering players' goals given by *mean-payoff* utility functions—arguably the most widely used approach for quantitative objectives in concurrent and multiagent systems. Finally, we consider the problem of computing outcomes that satisfy social welfare constraints. To this end, we consider both utilitarian and egalitarian social welfare and show that computing such outcomes is either PSPACE-complete or NP-complete.

Keywords Temporal logic · Game theory · Rational verification · Multi-agent systems

✉ Julian Gutierrez
julian.gutierrez@monash.edu

Muhammad Najib
najib@cs.uni-kl.de

Giuseppe Perelli
perelli@di.uniroma1.it

Michael Wooldridge
michael.wooldridge@cs.ox.ac.uk

[1] Faculty of Information Technology, Monash University, Melbourne, Australia

[2] Department of Computer Science, University of Kaiserslautern, Kaiserslautern, Germany

[3] Department of Computer Science, Sapienza University of Rome, Rome, Italy

[4] Department of Computer Science, University of Oxford, Oxford, UK

1 Introduction

The formal verification of computer systems has been a major research area in computer science for the past 60 years. Verification is the problem of checking program correctness: the key decision problem relating to verification is that of establishing whether or not a given system P satisfies a given specification. The most successful contemporary approach to formal verification is model checking, in which an abstract, finite state model of the system of interest P is represented as a Kripke structure K_P (a labelled transition system), and the specification is represented as a temporal logic formula φ, the models of which are intended to correspond to "correct" behaviours of the system [11]. The verification process then reduces to establishing whether the specification formula φ is satisfied in the Kripke structure K_P (notation: $K_P \models \varphi$), a process that can be efficiently automated in many settings of interest [7]. For example, model checking Linear Temporal Logic (LTL) specifications can be done in polynomial space, and for specifications in Computation Tree Logic (CTL) it can be done in polynomial time [8].

In the context of multiagent systems, *rational verification* forms a natural counterpart of model checking [16, 17, 33]. This is the problem of checking whether a given property φ, expressed as a temporal logic formula, is satisfied in a computation of a system that might be generated *if agents within the system choose strategies for selecting actions that form a game-theoretic equilibrium*. This game theoretic aspect of rational verification adds a new ingredient to the verification problem, as it becomes necessary to take into account the *preferences* of players with respect to the possible runs of the system. Typically, in rational verification, such preferences are given by associating an LTL goal γ_i with each player i in the game: player i prefers all those runs of the system that satisfy γ_i over those that do not, is indifferent between all those runs that satisfy γ_i, and is similarly indifferent between those runs that do not satisfy γ_i. In this setting, rational verification with respect to a specification φ is 2EXPTIME-complete, regardless of whether the representation of the system is given succinctly [16, 17] or explicitly simply as a finite-state labelled transition graph [15]. This high computational complexity represents a key barrier to the wider take-up of rational verification.

Our aim in this work is to improve this state of affairs: we present a range of settings for which we are able to give complexity results that greatly improve on the 2EXPTIME-complete result of the general LTL case. We first consider games where the goals of players are represented as GR(1) formulae. GR(1) is an important fragment of LTL that can express a wide range of practically useful response properties of concurrent and reactive systems [4]. We then consider *mean-payoff utility functions*: one of the most studied reward and quality measures used in games for automated formal verification. In each case, we study the rational verification problem for system specifications φ given as GR(1) formulae and as LTL formulae, with respect to system models that are formally represented as concurrent game structures [1].

Our main results, summarised in Table 1, show that in the cases mentioned above, the 2EXPTIME result can be dramatically improved, to settings where rational verification can be solved in polynomial space, NP, or even in polynomial time, if the number of players in the game is assumed to be fixed.

In addition to characterising the complexity of the core rational verification problems for these settings, we also consider the problem of computing strategy profiles for players that *maximise social welfare*. Measures of social welfare are measures of how well society as a whole fares with some particular game outcome; thus social welfare measures are

Table 1 Summary of main complexity results

Players' goals	Specification	E-Nash	
LTL	LTL	2EXPTIME-complete	
GR(1)	LTL	PSPACE-complete	(Corollary 1)
GR(1)	GR(1)	FPT	(Theorem 3)
mp	LTL	PSPACE-complete	(Corollary 2)
mp	GR(1)	NP-complete	(Theorem 5)

aggregate measures of utility. We look at two well-known measures of social welfare: *utilitarian social welfare* (in which we aim to maximise the sum of individual agent utilities) and *egalitarian social welfare* (in which we try to maximise the utility of the worst-off player). We show that, for mean payoff games, computing outcomes for these measures with LTL specifications is PSPACE-complete.

Related work The rational verification problem has been studied for a number of different settings, including iterated Boolean games, reactive modules games, and concurrent game structures [15–18]. In each of these settings, the main rational verification problems are 2EXPTIME-complete, and hence highly intractable. Rational verification is closely related to rational synthesis, which is also 2EXPTIME-complete both in the Boolean case [13] and with rational environments [24]. One might mitigate the problem of intractability by considering low-level languages such as omega-regular specifications [10, 31] and turn-based setting [9]. All of the above cases only consider perfect information. In settings with imperfect information, the problem has been shown to be undecidable both for games with succinct and explicit model representations [12, 22].

Our work also relates to LTL and mean-payoff (mp) games in general. While the former are already 2EXPTIME-complete even for two-player games (and in fact already 2EXPTIME-hard for many LTL fragments [2]), the latter are NP-complete for multi-player games [32] and in NP ∩co NP for two-player games [34], and in fact solvable in quasipolynomial time since they can be reduced to two-player perfect-information parity games [6]. Even though we provide several complexity results that improve on the complexity of the general case, our solutions are unlikely to run in polynomial time, for instance as CTL model checking, since rational verification subsumes problems that are typically not known to be solvable in polynomial time, such as model checking or automated synthesis with temporal logic specifications.

2 Preliminaries

Linear temporal logic LTL extends propositional logic with two operators, **X** ("next") and **U** ("until"), for expressing properties of paths [11, 27]. The syntax of LTL is defined with respect to a set AP of atomic propositions as follows:

$$\phi ::= \top \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathbf{X}\phi \mid \phi \mathbf{U} \phi$$

where $p \in$AP. As usual, we define $\phi_1 \wedge \phi_2 \equiv \neg(\neg\phi_1 \vee \neg\phi_2)$, $\phi_1 \rightarrow \phi_2 \equiv \neg\phi_1 \vee \phi_2$, $\mathbf{F}\phi \equiv \top \mathbf{U} \phi$, and $\mathbf{G}\phi \equiv \neg\mathbf{F}\neg\phi$. We interpret LTL formulae with respect to pairs (α,t), where

$\alpha \in (2^{\mathrm{AP}})^{\omega}$ is an infinite sequence of sets of atomic proposition that indicates which propositional variables are true in every time point and $t \in \mathbb{N}$ is a temporal index into α. As usual, by $\alpha_t \in 2^{\mathrm{AP}}$ we denote the t-th element of the infinite sequence α. Formally, the semantics of LTL is given by the following rules:

$$
\begin{array}{lll}
(\alpha, t) \vDash \top & & \\
(\alpha, t) \vDash p & \text{iff} & p \in \alpha_t \\
(\alpha, t) \vDash \neg\phi & \text{iff} & \text{it is not the case that } (\alpha, t) \vDash \phi \\
(\alpha, t) \vDash \phi \vee \psi & \text{iff} & (\alpha, t) \vDash \phi \text{ or } (\alpha, t) \vDash \psi \\
(\alpha, t) \vDash \mathbf{X}\phi & \text{iff} & (\alpha, t+1) \vDash \phi \\
(\alpha, t) \vDash \phi \mathbf{U} \psi & \text{iff} & \text{for some } t' \geq t : \big((\alpha, t') \vDash \psi \text{ and} \\
 & & \quad \text{for all } t \leq t'' < t' : (\alpha, t'') \vDash \phi\big).
\end{array}
$$

If $(\alpha,0) \vDash \phi$, we write $\alpha \vDash \phi$ and say that α *satisfies* ϕ.

General reactivity of rank 1 The language of *General Reactivity of rank 1*, (GR(1)), is the fragment of LTL containing formulae that are written in the following form [4]:

$$(\mathbf{GF}\psi_1 \wedge \ldots \wedge \mathbf{GF}\psi_m) \rightarrow (\mathbf{GF}\phi_1 \wedge \ldots \wedge \mathbf{GF}\phi_n),$$

where subformulae ψ_i and ϕ_i are Boolean combinations of atomic propositions.

Mean-payoff value For an infinite sequence $\beta \in \mathbb{R}^{\omega}$ of real numbers, let $\mathrm{mp}(\beta)$ be denote *mean-payoff* value of β, that is,

$$\mathsf{mp}(\beta) = \lim_{n \to \infty} \inf \mathsf{avg}_n(\beta)$$

where, for $n \in \mathbb{N}$, we define

$$\mathsf{avg}_n(\beta) = \frac{1}{n} \sum_{j=0}^{n-1} \beta_j.$$

Arenas An *arena* is a tuple

$$A = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}, s_0, \mathsf{tr}, \lambda \rangle$$

where N, Ac, and St are finite non-empty sets of *players* (write $N = |\mathrm{N}|$), *actions*, and *states*, respectively; $s_0 \in \mathrm{St}$ is the *initial state*; $\mathsf{tr} : \mathrm{St} \times \overrightarrow{Ac} \rightarrow \mathrm{St}$ is a *transition function* mapping each pair consisting of a state $s \in \mathrm{St}$ and an *action profile* $\mathrm{a} \in \mathbf{Ac} = \mathrm{Ac}^{\mathrm{N}}$, one for each player, to a successor state; and $\lambda : \mathrm{St} \rightarrow 2^{\mathrm{AP}}$ is a *labelling function*, which maps every state to a subset of *atomic propositions*—the atomic propositions that are true at that state.

We sometimes refer to an action profile $\vec{a} = (\mathsf{a}_1, \ldots, \mathsf{a}_n) \in \overrightarrow{Ac}$ as a *decision*, and denote by a_i the action taken by player i. We also consider *partial* decisions. For a set of players $C \subseteq \mathrm{N}$ and action profile a, we let a_C and a_{-C} be two tuples of actions, respectively, one for all players in C and one for all players in $\mathrm{N} \setminus C$. We also write a_i for $\mathrm{a}_{\{i\}}$ and a_{-i} for $\mathrm{a}_{\mathrm{N} \setminus \{i\}}$. For two decisions a and a', we write $(\mathsf{a}_C, \mathsf{a}'_{-C})$ to denote the decision where the actions for players in C are taken from a and the actions for players in $\mathrm{N} \setminus C$ are taken from a'.

A *path* $\pi = (s_0, \mathsf{a}^0), (s_1, \mathsf{a}^1), \ldots$ is an infinite sequence in $(\mathrm{St} \times \mathbf{Ac})^{\omega}$ such that $\mathsf{tr}(s_k, \mathsf{a}^k) = s_{k+1}$ for all k. Paths are generated in the arena by each player i selecting a *strategy* σ_i that will define how to make choices over time. We model strategies as finite state machines with output. Formally, for arena A, a strategy $\sigma_i = (Q_i, q_i^0, \delta_i, \tau_i)$ for player

i is a finite state machine with output (a transducer), where Q_i is a finite and non-empty set of *internal states*, q_i^0 is the *initial state*, $\delta_i : Q_i \times \mathbf{Ac} \to Q_i$ is a deterministic *internal transition function*, and $\tau_i : Q_i \to \mathrm{Ac}_i$ an *action function*, $\mathrm{Ac}_i \subseteq \mathrm{Ac}$ for all $i \in \mathrm{N}$. Let Str_i be the set of strategies for player i. A *strategy profile* $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is a vector of strategies, one for each player. As with actions, $\boldsymbol{\sigma}_i$ denotes the strategy assigned to player i in profile $\boldsymbol{\sigma}$. Moreover, by $(\boldsymbol{\sigma}_B, \boldsymbol{\sigma}'_C)$ we denote the combination of profiles where players in disjoint B and C are assigned their corresponding strategies in $\boldsymbol{\sigma}$ and $\boldsymbol{\sigma}'$, respectively.

Once a state s and a strategy profile $\boldsymbol{\sigma}$ are fixed, the game has an *outcome*, a path in A, which we denote by $\pi(\boldsymbol{\sigma},s)$. Because strategies are deterministic, $\pi(\boldsymbol{\sigma},s)$ is the unique path induced by $\boldsymbol{\sigma}$, that is, the sequence $(s_0, \mathsf{a}^0), (s_1, \mathsf{a}^1), \ldots$ such that

- $s_{k+1} = \mathrm{tr}(s_k, \vec{a}_k)$, and
- $\vec{a}_{k+1} = (\tau_1(q_1^k), \ldots, \tau_n(q_n^k))$, for all $k \geq 0$.

Where $q_i^{k+1} = \delta_i(q_i^k, (\tau_1(q_1^k), \ldots, \tau_n(q_n^k)))$ is the unique sequence of internal states of strategy σ_i in $\boldsymbol{\sigma}$ obtained by feeding the result of previous computation at each step.

Arenas define the dynamic structure of games (the actions that agents can perform and their consequences), but lack the feature of games that gives them their strategic nature: players' preferences. A *multi-player game* is obtained from an arena A by associating each player with a *goal*. As indicated above, previous work has considered players with goals expressed as LTL formulae, with the idea being that an agent will act as best they can to ensure their LTL goal is satisfied (taking into account the fact that other players will act likewise). In the present article, we consider both goals that are expressed as GR(1) formulae, and mean payoff (mp) goals:

- A multi-player GR(1) game is a tuple $\mathcal{G}_{\mathsf{GR(1)}} = \langle A, (\gamma_i)_{i \in \mathrm{N}} \rangle$ where A is an arena and γ_i is the GR(1) goal for player i.
- A multi-player mp game is a tuple $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$, where A is an arena and $\mathsf{w}_i : \mathrm{St} \to \mathbb{Z}$ is a function mapping every state of the arena into an integer.

When it is clear from the context, we refer to a multi-player GR(1) or mp game as a *game* and denote it by $\mathcal{G}$. In any game with arena A, a path π in A induces a sequence $\lambda(\pi) = \lambda(s_0)\lambda(s_1)\cdots$ of sets of atomic propositions; if, in addition, A is the arena of an mp game, then, for each player i, the sequence $\mathrm{w}_i(\pi) = \mathrm{w}_i(s_0)\mathrm{w}_i(s_1)\cdots$ of weights is also induced.

For a GR(1) game and a path π in it, the payoff of a player i is $\mathrm{pay}_i(\pi) = 1$ if $\lambda(\pi) \vDash \gamma_i$ and $\mathrm{pay}_i(\pi) = 0$ otherwise. Regarding an mp game, the payoff of player i is $\mathrm{pay}_i(\pi) = \mathrm{mp}(\mathrm{w}_i(\pi))$. Moreover, for a GR(1) game and a path π, by $\mathrm{Win}(\pi) = \{i \in \mathrm{N} : \lambda(\pi) \vDash \gamma_i\}$ and $\mathrm{Lose}(\pi) = \{j \in \mathrm{N} : \lambda(\pi) \nvDash \gamma_j\}$ we denote the set of *winners* and *losers*, respectively, over π, that is, the set of players that get their goal satisfied and not satisfied, respectively, over π. With an abuse of notation, we sometime denote $\mathrm{Win}(\boldsymbol{\sigma},s) = \mathrm{Win}(\pi(\boldsymbol{\sigma},s))$ and $\mathrm{Lose}(\boldsymbol{\sigma},s) = \mathrm{Lose}(\pi(\boldsymbol{\sigma},s))$, respectively, the set of winners and losers over the path generated by strategy profile $\boldsymbol{\sigma}$ when starting the game from s. Furthermore, we simply write $\pi(\boldsymbol{\sigma})$ for $\pi(\boldsymbol{\sigma},s_0)$.

Nash equilibrium Using payoff functions, we can define the concept of Nash equilibrium [25]. For a game $\mathcal{G}$, a strategy profile $\boldsymbol{\sigma}$ is a *Nash equilibrium* of $\mathcal{G}$ if, for every player i and strategy $\sigma'_i \in \mathrm{Str}_i$, we have

$$\mathsf{pay}_i(\pi(\boldsymbol{\sigma})) \geq \mathsf{pay}_i(\pi((\boldsymbol{\sigma}_{-i}, \sigma'_i))) .$$

Let NE($\mathcal{G}$) be the set of Nash equilibria of $\mathcal{G}$.

E-Nash and rational verification In rational verification, a key question/problem is E-Nash, which is concerned with the existence of a Nash equilibrium that fulfils a given temporal specification φ. Formally, E-Nash is defined as follows:

Definition 1 (E-Nash) Given a game $\mathcal{G}$ and a formula φ:
Does there exist $\boldsymbol{\sigma} \in \text{NE}(\mathcal{G})$ such that $\pi(\boldsymbol{\sigma}) \vDash \varphi$?

Previous work [15–18] has demonstrated that, if we assume player goals are expressed as LTL formulae, the E-Nash problem is 2EXPTIME-complete, and hence highly intractable. Motivated by this, in this article, we study E-Nash for a number of relevant instantiations of the problem, which we show to have better (lower) computational complexity. In particular, we study cases where

- Specifications φ are LTL and players' goals are GR(1);
- Specifications φ are LTL and players have mp goals;
- Both the specification φ and the goals are GR(1);
- Specifications φ are GR(1) and players have mp goals.

Automata Some of the algorithms we present for the E-Nash problem use techniques from automata theory. Specifically, we use deterministic automata on infinite words with *Streett* acceptance conditions. Formally, a *deterministic Streett automaton on infinite words* (DSW) is a tuple $\mathcal{A} = (\Sigma, Q, q^0, \delta, \Omega)$ where Σ is the input alphabet, Q is a finite set of states, $\delta : Q \times \Sigma \to Q$ is a transition function, q^0 is an initial state, and Ω is a Streett acceptance condition. A Streett condition Ω is a set of pairs $\{(E_1, C_1), \ldots, (E_n, C_n)\}$ where $E_k \subseteq Q$ and $C_k \subseteq Q$ for all $k \in [1,n]$. A run ρ is accepting in a DSW $\mathcal{A}$ with condition Ω if ρ either visits E_k finitely many times or visits C_k infinitely often, i.e., if for every k either $inf(\rho) \cap E_k = \emptyset$ or $inf(\rho) \cap C_k \neq \emptyset$.

3 Games of general reactivity of rank 1

We consider two variations of GR(1) games: in the first, the specification formula is expressed in LTL, while the goals are in GR(1); in the second, both the specification formula and the goals belong to GR(1). We begin by providing a general result characterizing Nash Equilibrium for GR(1), which is given in terms of *punishments*. We first require some notation.

For a GR(1) game $\mathcal{G}$, player $j \in$N, and state $s \in$St, the strategy profile $\boldsymbol{\sigma}_{-j}$ is *punishing* for player j in s if $\pi((\boldsymbol{\sigma}_{-j}, \sigma'_j), s) \nvDash \gamma_j$, for every possible strategy σ'_j of player j. We say that a state s is punishing for j if there exists a punishing strategy profile for j on s. Moreover, we denote by $\text{Pun}_j(\mathcal{G})$ the set of punishing states in $\mathcal{G}$. A pair $(s,\text{a}) \in \text{St} \times \mathbf{Ac}$ is *punishing-secure* for player j, if $\text{tr}(s, (\text{a}_{-j}, \text{a}'_j)) \in \text{Pun}_j(\mathcal{G})$ for every action a'_j.

Theorem 1 In a given GR(1) game $\mathcal{G}$, there exists a Nash Equilibrium if and only if there exists an ultimately periodic path π such that, for every $k \in \mathbb{N}$, the pair (s_k, a^k) of the k-th iteration of π is punishing-secure for every $j \in \text{Lose}(\pi)$.

Proof Proof sketch The proof proceeds by double implication.

From left to right, let $\boldsymbol{\sigma} \in \mathrm{NE}(\mathcal{G})$ and π be the ultimately periodic path generated by $\boldsymbol{\sigma}$. Assume by contradiction that π is not punishing-secure for some $j \in \mathrm{Lose}(\pi)$, that is, there is $k \in \mathbb{N}$ and action a'_j such that $\mathsf{tr}(s_k, (\mathsf{a}_{-j}, \mathsf{a}'_j)^k) \notin \mathrm{Pun}_j(\mathcal{G})$. Thus, j can deviate at s_k and satisfy γ_j, which is a contradiction to $\boldsymbol{\sigma}$ being a Nash equilibrium.

From right to left, recall that π can be generated by a finite transducer, say $\mathcal{T}^\pi = \langle T, t_0, \delta^\pi, \tau^\pi \rangle$ with $\delta^\pi : T \times \mathbf{Ac} \to T$ being the internal function and $\tau^\pi : T \to \mathbf{Ac}$ being the action function that generates π. Moreover, observe that such transducer can be decomposed into strategies $\sigma_i^\pi = \langle T, t_0, \delta^\pi, \tau_i^\pi \rangle$ where $\tau_i^\pi(t) = \tau^\pi(t)_i$. Moreover, for every losing player $j \in \mathrm{Lose}(\pi)$, there is a memoryless punishing strategy profile $\sigma_{-j}^{\mathsf{pun}} : \mathrm{St} \to \mathbf{Ac}_{-j}$ for j in every $s \in \mathrm{Pun}_j(\mathcal{G})$. Such strategy can also be decomposed and *distributed* to the agents different from j as $\sigma_{-j}^{\mathsf{pun},i}(s) = \sigma_{-j}^{\mathsf{pun}}(s)_i$ for every $i \in \mathrm{N} \setminus \{j\}$.

Now, for every agent i, consider the strategy $\sigma_i = \langle Q_i, q_i^0, \delta_i, \tau_i \rangle$ defined as follows:

- $Q_i = T \times S \times (\{\top\} \cup \mathrm{Lose}(\pi))$;
- $q_i^0 = (t_0, s_0, \top)$;
- δ_i is defined as follows:[1]

$$\delta_i(t, s, \top, \mathsf{a}) = \begin{cases} (\delta^\pi(t, \mathsf{a}), \mathsf{tr}(s, \mathsf{a}), \top), & \text{if } \mathsf{a} = \tau^\pi(t) \\ (\delta^\pi(t, \mathsf{a}), \mathsf{tr}(s, \mathsf{a}), j), & \text{if } \mathsf{a}_{-j} = (\tau^\pi(t))_{-j} \text{ and } \mathsf{a}_j \neq (\tau^\pi(t))_j \end{cases}$$

$$\delta_i(t,s,j,\mathrm{a}) = (\delta^\pi(t,\mathrm{a}),\mathrm{tr}(s,\mathrm{a}),j)$$

- $\tau_i(t, s, \iota) = \begin{cases} \tau_i^\pi(t) & \text{if } \iota = \top \\ \sigma_{-\iota}^{\mathsf{pun},i}(s) & \text{otherwise} \end{cases}$

Intuitively, the strategy σ_i mimics the transducer $\mathcal{T}^\pi$ to produce the play π. In addition to this, it keeps track of the actions taken by the losing agents, checking whether they adhere to the transducer or they deviate unilaterally from it. In case of a deviation of agent j, the strategy σ_i flags the deviating agent and switches from mimicking $\mathcal{T}^\pi$ to adopting the punishment strategy σ_j^{pun}.

We need to show that the strategy profile $\boldsymbol{\sigma}$ is a Nash Equilibrium.

Clearly, as $\pi(\boldsymbol{\sigma}) = \pi$, all the agents that are winning over π do not have a beneficial deviation. For a losing agent j, observe that a unilateral deviation σ_j triggers the strategy profile $\boldsymbol{\sigma}_{-j}$ to implement a punishment over j. Moreover, observe that GR(1) objectives are prefix-independent, which implies that the punishment takes effect no matter at which instant of the computation is started being adopted. Therefore, every deviation $\boldsymbol{\sigma}'_j$ cannot be beneficial for agent j, and hence $\boldsymbol{\sigma}$ is a Nash Equilibrium. □

With this result in place, the following procedure can be seen to solve E-Nash:

1. Guess a set $W \subseteq \mathrm{N}$ of winners;
2. For each player $j \in L = \mathrm{N} \setminus W$, a loser in the game, compute its punishment region $\mathrm{Pun}_j(\mathcal{G})$;

[1] Note that we should define the internal and action functions on their entire domains. However, their definition for the other cases is irrelevant in the proof.

3. Remove from $\mathcal{G}$ the states that are not punishing for players $j \in L$ and the edges (s, s') that are labelled with an action profile a such that (s,a) is not punishing-secure for some $j \in L$, thus obtaining a game $\mathcal{G}^{-L}$;
4. Check whether there exists an ultimately periodic path π in $\mathcal{G}^{-L}$ such that $\pi \vDash \varphi \wedge \bigwedge_{i \in W} \gamma_i$ holds.

Expressed more formally, the above procedure yields Algorithm 1.

Algorithm 1: E-Nash of GR(1) games.

```
1 Input: A game G_GR(1) and a specification formula φ.
2 for i ∈ N do
3     Compute Pun_i(G)
4 for W ⊆ N do
5     Compute L = N \ W
6     Compute G^{-L}
7     if π ⊨ (φ ∧ ⋀_{i∈W} γ_i) for some π ∈ G^{-L} then
8         return Accept
9 return Reject
```

While line 6 requires solving the model checking problem for an LTL formula, which can be done in polynomial space, line 5 can be done in polynomial time. Line 4, on the other hand, makes the procedure run in exponential time in the number of players, but still in polynomial space. We then only need to consider line 3: this step can be done in polynomial time, as we now show.

Theorem 2 For a given GR(1) game $\mathcal{G}$ over the arena $A = \langle \mathrm{N},\mathrm{Ac},\mathrm{St},s_0,\mathrm{tr},\lambda \rangle$ and a player $j \in \mathrm{N}$, computing the punishing region $\mathrm{Pun}_j(\mathcal{G})$ of player j can be done in polynomial time with respect to the size of both $\mathcal{G}$ and γ_j.

Proof We reduce the problem to computing the winning region of a suitably defined Streett game with a single pair as the winning condition, whose complexity is known to be $O(mn^{k+1}kk!)$ [26]. Given that in our case we have $k = 1$, we obtain a polynomial time algorithm.

Recall that the goal of player j is of the form:

$$\gamma_j = \bigwedge_{l=1}^{m_j} \mathbf{GF}\psi_l^j \rightarrow \bigwedge_{r=1}^{n_j} \mathbf{GF}\theta_r^j,$$

where ψ_l^j's and θ_r^j's are boolean combinations of atomic propositions. Then, consider the arena $A' = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}', s_0', \mathsf{tr}' \rangle$[2] where

- $\mathrm{St}' = \mathrm{St} \times \{0, \ldots, m_j\} \times \{0, \ldots, n_j\}$;
- $s_0' = (s_0, 0, 0)$;
- $\mathsf{tr}'((s, \iota_1, \iota_2), \mathrm{a}) = (\mathsf{tr}(s, \mathrm{a}), \iota_1', \iota_2')$ where

[2] We omit the definition of labelling function, as not needed here.

$$\iota_1' = \begin{cases} (\iota_1 \oplus_{(m_j+1)} 1), & \text{if } \iota_1 = 0 \text{ or } s \vDash \psi^j_{\iota_1}. \\ \iota_1, & \text{otherwise.} \end{cases}$$

$$\iota_2' = \begin{cases} (\iota_2 \oplus_{(n_j+1)} 1), & \text{if } \iota_2 = 0 \text{ or } s \vDash \theta^j_{\iota_2}. \\ \iota_2, & \text{otherwise.} \end{cases}$$

And by $\oplus_k$ we denote the addition modulo k.

Intuitively, arena A' mimics the behaviour of A and carries two indexes, ι_1 and ι_2. Index ι_1 is increased by one every time the path visits a state that satisfies $\psi^j_{\iota_1}$ and resets to 0 every time the path visits a state that satisfies $\psi^j_{m_j}$. Clearly, ι_1 is reset infinitely many times if and only if the path satisfies every ψ^j_l infinitely many times, and so if and only if it satisfies the temporal specification $\bigwedge_{l=1}^{m_j} \mathbf{GF}\psi^j_l$. The same argument applies to index ι_2, but with respect to the boolean combinations θ^j_r's.

Now, consider the sets $C_j = \text{St} \times \{0\} \times \{0,\ldots,n_j\}$ and $E_j = \text{St} \times \{0,\ldots,m_j\} \times \{0\}$. Clearly, the Streett pair (C_j,E_j) is satisfied by all and only the paths in A' that satisfy γ_j. Therefore, the winning region of γ_j can be computed as the winning set of the Streett game with (C_j,E_j) being the only Streett pair. Observe that the winning region is computable as Street games are *determined*. Moreover, having a number of pairs fixed, the computation can be done in polynomial time, which proves our statement. □

Based on Theorem 2, we have the following result.

Corollary 1 The E-Nash problem for GR(1) games with an LTL specification is PSPACE-complete.

Proof The upper-bound follows from the procedure described above. Regarding the lower-bound, note that model-checking an LTL formula φ against a Kripke structure $\mathcal{K}$ can be easily encoded as an instance of E-Nash where $\mathcal{G}$ is played over a Kripke structure $\mathcal{K}$, taken to be its arena, players' goals being tautologies, and the specification being $\neg\varphi$. In such a case, we have that $\mathcal{K} \vDash \varphi$ if and only if E-Nash for the pair $(\mathcal{G}, \varphi)$ has a negative answer. □

Corollary 1 sharply contrasts with the complexity of E-Nash when goals expressed as LTL formulae: in this more general case, E-Nash is 2EXPTIME-complete.

The special case of GR(1) specifications One of hardest parts of Algorithm 1 is line 6, where an LTL model checking problem must be solved, thereby making the running time of the overall procedure exponential in the size of the specification and goals of the players. As we show in the reminder of this section, one way to drastically reduce the complexity of our decision procedure is to require that the specification is also expressed in GR(1). In such a case, the LTL model checking procedure in line 6 of Algorithm 1 can be avoided, leading to a much simpler construction, which runs in polynomial time for every fixed number of players. In this section, we provide precisely such a simpler construction.

Recall that every GR(1) specification φ can be regarded as a Streett condition with a single pair over an arena A' suitably constructed from the original arena A [3]. Thus, by denoting (C_φ,E_φ) and (C_i,E_i) the Streett pairs corresponding to the GR(1) conditions φ and γ_i, respectively, the problem of finding a path in A' satisfying the formula $\varphi \wedge \bigwedge_{i \in W} \gamma_i$

amounts to deciding the emptiness of the Streett automaton $\mathcal{A} = \langle \mathbf{Ac}, \mathrm{St}', s_0', \mathsf{tr}, \Omega \rangle$ where $\Omega = \{(C_\varphi, E_\varphi), (C_{\gamma_i}, E_{\gamma_i})_{i \in W}\}$.

Note that the size of A' is polynomial in the size of the GR(1) formulae involved, polynomial in the number of states and actions in the original arena A, and exponential in the number of players. More specifically, we have that $|\mathrm{St}'| = |\mathrm{St}| \cdot |\gamma|^{|\mathrm{N}|}$ and so the number of edges is at most $|\mathrm{St}'|^2$. Moreover, the emptiness problem of a deterministic Streett word automaton can be solved in time that is polynomial in the automaton's index and its number of states and transitions [23, 29]. The complexity of the E-Nash problem takes $2^{|\mathrm{N}|}$ times a procedure for computing at most $|N|$ punishing regions (that is polynomial in the size of both $\mathcal{G}$ and $\varphi,\gamma_1,\ldots,\gamma_N$) plus the complexity of the emptiness problem for a Streett automaton whose size is polynomial in $\mathcal{G}$ $\varphi,\gamma_1,\ldots,\gamma_N$, and exponential in the number of players.

Based on the constructions described above, we have the following (fixed-parameter tractable) complexity result.

Theorem 3 For a given GR(1) game $\mathcal{G}$ and a GR(1) formula φ, the E-Nash problem can be solved in time that is polynomial in |St|, |Ac|, and $|\varphi|, |\gamma_1|,\ldots,|\gamma_N|$ and exponential in the number of players |N|. Therefore, the problem is fixed-parameter tractable, parametrized in the number of players.

4 Mean-payoff games

We now focus on multi-player mean-payoff (mp) games. As in the previous case, we first characterise the Nash Equilibria of a game in terms of punishments and then reduce E-Nash to a suitable path-finding problem in the underlying arena. To do this, we first need to recall the notion of secure values for mean-payoff games [32].

For a player i and a state $s \in$St, by $\mathsf{pun}_i(s)$ we denote the punishment value of i over s, that is, the maximum payoff that i can achieve from s, when all other players behave adversarially. Such a value can be computed by considering the corresponding two-player zero-sum mean-payoff game [34]. Thus, it is in NP $\cap$co NP, and note that both player i and coalition $\mathrm{N} \setminus \{i\}$ can achieve the optimal value of the game using memoryless strategies.

For a player i and a value $z \in \mathbb{R}$, a pair (s,a) is z-secure for i if $\mathsf{pun}_i(\mathsf{tr}(s, (\mathsf{a}_{-i}, \mathsf{a}_i'))) \leq z$ for every $\mathsf{a}_i' \in$ Ac.

Theorem 4 For every mp game $\mathcal{G}$ and ultimately periodic path $\pi = (s_0,\mathrm{a}_0),(s_1,\mathrm{a}_1),\ldots$, the following are equivalent

1. There is $\boldsymbol{\sigma} \in \mathrm{NE}(\mathcal{G})$ such that $\pi = \pi(\boldsymbol{\sigma},s_0)$;
2. There exists $\mathbf{z} \in \mathbb{R}^{\mathrm{N}}$, where $z_i \in \{\mathsf{pun}_i(s) : s \in \mathrm{St}\}$ such that, for every $i \in$N
   (a) for all $k \in \mathbb{N}$, the pair (s_k, a^k) is z_i-secure for i, and
   (b) $z_i \leq \mathsf{pay}_i(\pi)$.

Proof The proof proceeds by double implication.

For the case (1) $\Rightarrow$ (2), assume that $\boldsymbol{\sigma} \in \mathrm{NE}(\mathcal{G})$ is such that $\pi(\boldsymbol{\sigma}) = \pi$. Thus, define $z_i = \max\{\mathsf{pun}_i(\mathsf{tr}(s_k, (\mathsf{a}^k_{-i}, \mathsf{a}'))) : k \in \mathbb{N}, \mathsf{a}' \in \mathrm{Ac}_i\}$, that is, the max value agent i can achieve by unilaterally deviating from any point in π and getting immediately punished. By definition, we obtain that (s_k, a^k) is z_i-secure for i, at every $k \in \mathbb{N}$. Moreover, assume by

contradiction that $\mathsf{pay}_i(\pi) < z_i$ for some agent i. Then, let $k \in \mathbb{N}$ and $\mathsf{a}'_i \in \mathrm{Ac}_i$ be such that $z_i = \mathsf{pun}_i(s_k, (\mathsf{a}_{-i}, \mathsf{a}'_i))$. Thus, there exists a strategy σ'_i that follows $\boldsymbol{\sigma}_i$ for k steps and then deviates using a'_i that ensures a payoff of z_i for agent i. Such strategy is a beneficial deviation of agent i from $\boldsymbol{\sigma}$, in contradiction with the fact that $\boldsymbol{\sigma}$ is a Nash Equilibrium.

For the case (2) $\Rightarrow$ (1), we define a strategy profile $\boldsymbol{\sigma}$ and then prove it is a Nash Equilibrium. First observe that, being π ultimately periodic, there exists a finite transducer $\mathcal{T}^\pi = \langle T, t_0, \delta^\pi, \tau^\pi \rangle$ with $\delta^\pi : T \times \mathbf{Ac} \to T$ being the internal function and $\tau^\pi : T \to \mathbf{Ac}$ being the action function that generates π. Moreover, observe that such transducer can be decomposed into strategies $\sigma^\pi_i = \langle T, t_0, \delta^\pi, \tau^\pi_i \rangle$ where $\tau^\pi_i(t) = \tau^\pi(t)_i$. In addition to this, for every agent j, consider the *memoryless* strategy $\sigma^{\mathsf{pun}}_{-j} : \mathrm{St} \to \mathbf{Ac}_{-j}$ that minimizes the payoff of agent j in every state $s \in$St. Such strategy can also be decomposed and *distributed* to the agents different from j as $\sigma^{\mathsf{pun},i}_{-j}(s) = \sigma^{\mathsf{pun}}_{-j}(s)_i$ for every $i \in \mathrm{N} \setminus \{j\}$. Now, for every agent i, consider the strategy $\sigma_i = \langle Q_i, q^0_i, \delta_i, \tau_i \rangle$ defined as follows:

- $Q_i = T \times S \times (\{\top\} \cup \mathrm{N} \setminus \{i\})$;
- $q^0_i = (t_0, s_0, \top)$;
- δ_i is defined as follows:

$$\delta_i(t, s, \top, \mathsf{a}) = \begin{cases} (\delta^\pi(t, \mathsf{a}), \mathsf{tr}(s, \mathsf{a}), \top), & \text{if } \mathsf{a} = \tau^\pi(t) \\ (\delta^\pi(t, \mathsf{a}), \mathsf{tr}(s, \mathsf{a}), j), & \text{if } \mathsf{a}_{-j} = (\tau^\pi(t))_{-j} \text{ and } \mathsf{a}_j \neq (\tau^\pi(t))_j \end{cases}$$

$$\delta_i(t,s,j,\mathrm{a}) = (\delta^\pi(t,\mathrm{a}),\mathrm{tr}(s,\mathrm{a}),j)$$

- $\tau_i(t, s, \iota) = \begin{cases} \tau^\pi_i(t) & \text{if } \iota = \top \\ \sigma^{\mathsf{pun},i}_{-\iota}(s) & \text{otherwise} \end{cases}$ [3]

Intuitively, the strategy σ_i mimics the transducer $\mathcal{T}^\pi$ to produce the play π. In addition to this, it keeps track of the actions taken by the other agents, checking whether they adhere to the transducer or they deviate unilaterally from it. In case of a deviation of agent j, the strategy σ_i flags the deviating agent and switches from mimicking $\mathcal{T}^\pi$ to adopting the punishment strategy σ^{pun}_j. Clearly, the strategy profile $\boldsymbol{\sigma} = \langle \sigma 1, \ldots, \sigma_n \rangle$ is such that $\pi(\boldsymbol{\sigma}) = \pi$. It remains to show that it is a Nash Equilibrium. Note that, since every strategy σ_i adopts the punishment for agent j at every possible deviation. Note that, being $\overline{\mp}$ a prefix independent condition, the payoff for agent j is punished no matter at which instant the punishment strategy is started being adopted. At this point, being every pair (s_k, a^k) in π z_j-secure for agent j, it holds that every deviation of agent j does not ensure a payoff greater than z_j, that is $\mathsf{pay}_j(\boldsymbol{\sigma}_{-j}, \boldsymbol{\sigma}'_j) \leq z_j$. On the other hand, from condition (b) of item 2 in the statement, we have that $z_j \leq \mathsf{pay}_j(\boldsymbol{\sigma})$. By putting these two conditions together, we obtain

$$\mathsf{pay}_j(\boldsymbol{\sigma}_{-j}, \boldsymbol{\sigma}'_j) \leq z_j \leq \mathsf{pay}_j(\boldsymbol{\sigma}).$$

This proves that every deviation of agent j from $\boldsymbol{\sigma}$ is not beneficial, and so that $\boldsymbol{\sigma}$ is a Nash Equilibrium. □

The characterization of Nash Equilibria provided in Theorem 4 allows us to turn the E-Nash problem for mp games into a path finding problem over $\mathcal{G}$. Similarly to the case of GR(1) games, we have the following procedure.

[3] Note that weshould define the internal and action functions on their entire domains. However, their definition for the other cases is irrelevant in the proof.

1. For every $i \in$N and $s \in$St, compute the value $\text{pun}_i(s)$;
2. Guess a vector $z \in \mathbb{R}^{\text{N}}$ of values, each of them being a punishment value for a player i;
3. Compute the game $\mathcal{G}[z]$ by removing the states s such that $\text{pun}_i(s) \leq z_i$ for some player i and the transitions (s,a) that are not z_i secure for some player i;
4. Find an ultimately periodic path π in game $\mathcal{G}[z]$ such that $\pi \models \varphi$ and $z_i \leq \text{pay}_i(\pi)$ for every player $i \in$N.

Step 1 can be done in NP for every pair (i,s), step 2 can be done in exponential time and polynomial space in the number of z-secure values, and step 3 can be done in polynomial time, similar to the case of GR(1) games. Regarding the last step, its complexity depends on the specification language. For the case of φ being an LTL formula, consider the formula

$$\varphi_{\text{E-Nash}} := \varphi \wedge \bigwedge_{i \in \text{N}} (\text{mp}(i) \geq z_i),$$

written in the language LTL^{Lim}, an extension of LTL where statements about mean-payoff values over a given weighted arena can be made [5]. Observe that formula $\varphi_{\text{E-Nash}}$ corresponds exactly to requirement 2(b) in Theorem 4. Moreover, since every path in $\mathcal{G}[z]$ satisfies condition 2(a) by construction, every path that satisfies $\varphi_{\text{E-Nash}}$ is a solution of the E-Nash problem and *vice versa*. We can solve the latter problem by model checking the formula against the arena underlying $\mathcal{G}[z]$. Since this can be done in PSPACE [5], we have the following result.

Corollary 2 The E-Nash problem for mp games with an LTL specification formula φ is PSPACE-complete.

As for the case of GR(1) games, we can summarize the procedure in the following algorithm (Algorithm 2).

Algorithm 2: E-Nash of mp games.

```
Input: A game G_mp and a specification formula φ.
for i ∈ N and s ∈ St do
    Compute pun_i(G)
for z⃗ ∈ {pun_i(s) : s ∈ St}^N do
    Compute G[z]
    if π ⊨ φ_E-Nash for some π ∈ G[z] then
        return Accept
return Reject
```

The special case of GR(1) specifications As in the case of GR(1) games, here we show that restricting the specification language to GR(1) also lowers the complexity for mp games. The reason for this is that the path finding problem for GR(1) specifications can be done while avoiding model-checking an LTL^{Lim} formula. In order to do this, we follow a different approach. Using an mp game $\mathcal{G}$ and a GR(1) specification ϕ we define a linear program such that the linear program has a solution if and only if the pair $(\mathcal{G}, \phi)$ is an instance of E-Nash. In particular, this approach is similar to the technique used in [19, Theorem 2], where Linear Programming is used to find the complexity of solving a variant of E-Nash. Formally, we have the following result.

Theorem 5 The E-Nash problem for mp games with a GR(1) specification φ is NP-complete.

Proof We will define a linear program of size polynomial in $\mathcal{G}$ having a solution if and only if there exists an ultimately periodic path whose payoff for every player i is at least a minimum threshold z_i and satisfies the GR(1) specification.

In order to do that, first recall that φ has the following form

$$\varphi = \bigwedge_{l=1}^{m} \mathbf{GF}\psi_l \rightarrow \bigwedge_{r=1}^{n} \mathbf{GF}\theta_r,$$

and let $V(\psi_l)$ and $V(\theta_r)$ be the subset of states in $\mathcal{G}$ that satisfy the Boolean combinations ψ_l and θ_r, respectively. Observe that property φ is satisfied over a path π if, and only if, either π visits every $V(\theta_r)$ infinitely many times or visits some of the $V(\psi_l)$ only a finite number of times.

For the game $\mathcal{G}[z]$, let $\langle V, E, (\mathsf{w}'_i)_{i\in\mathrm{N}}\rangle$ be the underlying graph, where $\mathsf{w}'_i(v) = \mathsf{w}_i(v) - z_i$ for every $i \in$N, and $v \in V \subseteq$ St. Furthermore, for every edge $e \in E$, we introduce a variable x_e. Informally, the value x_e is the number of times that the edge e is used on a cycle. Formally, let:

- $\mathrm{src}(e) = \{v \in V : \exists w e = (v,w) \in E\}$;
- $\mathrm{trg}(e) = \{v \in V : \exists w e = (w,v) \in E\}$;
- $\mathrm{out}(v) = \{e \in E : \mathrm{src}(e) = v\}$;
- $\mathrm{in}(v) = \{e \in E : \mathrm{trg}(e) = v\}$.

Consider ψ_l for some $1 \leq l \leq m$, and define the linear program LP(ψ_l) with the following inequalities and equations:

Eq1: $x_e \geq 0$ for each edge e
a basic consistency criterion;

Eq2: $\Sigma_{e\in E} x_e \geq 1$
ensures that at least one edge is chosen;

Eq3: for each $i \in$N, $\Sigma_{e\in E} \mathsf{w}'_i(\mathsf{src}(e))x_e \geq 0$
ensures that the total sum of any solution is positive;

Eq4: $\Sigma_{\mathsf{src}(e)\cap V(\psi_l)\neq\emptyset} x_e = 0$
ensures that no state in $V(\psi_l)$ is in the cycle associated with the solution;

Eq5: for each $v \in V$, $\Sigma_{e\in\mathrm{out}(v)} x_e = \Sigma_{e\in\mathrm{in}(v)} x_e$
says that the number of times one enters a vertex is equal to the number of times one leaves that vertex.

By construction, it follows that LP(ψ_l) admits a solution if and only if there exists a path π in $\mathcal{G}$ such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i and visits $V(\psi_l)$ only *finitely many times*. Note that the condition $z_i \leq \mathrm{pay}_i(\pi)$ is ensured by Eq3. Indeed, the value of a path π in $\mathcal{G}[z]$ that is represented in a solution to LP(ψ_l), and thus satisfying Eq3, is such that $0 \leq \mathsf{pay}_i^{\mathcal{G}[z]}(\pi)$, with $pay_i^{\mathcal{G}[z]}$ representing the payoff function for agent i in the game $\mathcal{G}[z]$. Now observe that, as the weights in $\mathcal{G}[z]$ are all downshifted by a value z_i for every agent i, it holds that $\mathsf{pay}_i(\pi) = \mathsf{pay}_i^{\mathcal{G}[z]}(\pi) + z_i$, which in turns implies that $z_i \leq \mathrm{pay}_i(\pi)$.

Now, consider also the linear program LP(θ_1,…,θ_n) defined with the following inequalities and equations:

Eq1: $x_e \geq 0$ for each edge e
a basic consistency criterion;
Eq2: $\Sigma_{e \in E} x_e \geq 1$
ensures that at least one edge is chosen;
Eq3: for each $i \in$N, $\Sigma_{e \in E} \mathsf{w}'_i(\mathsf{src}(e)) x_e \geq 0$
ensures that the total sum of any solution is positive;
Eq4: for all $1 \leq r \leq n$, $\Sigma_{\mathsf{src}(e) \cap V(\theta_r) \neq \emptyset} x_e \geq 1$
ensures that for every $V(\theta_r)$ at least one state is in the cycle;
Eq5: for each $v \in V$, $\Sigma_{e \in \mathrm{out}(v)} x_e = \Sigma_{e \in \mathrm{in}(v)} x_e$
says that the number of times one enters a vertex is equal to the number of times one leaves that vertex.

In this case, LP(θ_1,…,θ_n) admits a solution if and only if there exists a path π such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i and visits every $V(\theta_r)$ *infinitely many times*.

Since the constructions above are polynomial in the size of both $\mathcal{G}$ and ϕ, we can conclude it is possible to check in NP the statement that there is a path π satisfying φ such that $z_i \leq \mathrm{pay}_i(\pi)$ for every player i in the game if and only if one of the two linear programs defined above has a solution. For the lower bound, we use [32] and observe that if ϕ is true, then the problem is equivalent to checking whether the mp game has a Nash equilibrium. □

5 Social welfare verification

Until this point, the problems considered primarily concerned about the satisfaction of a temporal logic property φ over the game $\mathcal{G}$. However, one might be interested in achieving an outcome that is somehow best also for the agent society. To capture this setting, we introduce *social welfare* measures. Social welfare measures are *aggregate* measures of utility. Thus, a social welfare measure takes as input a profile of utilities, one for each player in the game, and somehow aggregates these into an overall measure, indicating how good the outcome is for society as a whole. Note that since social welfare is inherently a quantitative measure, in this section we restrict our attention to mp games.

Formally, for a game $\mathcal{G}$ with a set N of agents, a *social welfare function* sw takes the form

$$\mathsf{sw} : \mathbb{R}^N \to \mathbb{R}$$

Thus, a social welfare function maps a N-tuple of real numbers into a real number which represents the *aggregated payoff*. More specifically, for a strategy profile $\boldsymbol{\sigma}$, the social welfare of $\boldsymbol{\sigma}$ is given by $\mathrm{sw}(\mathrm{pay}_1(\boldsymbol{\sigma}),\ldots,\mathrm{pay}_N(\boldsymbol{\sigma}))$. With an abuse of notation, we denote $\mathrm{sw}(\boldsymbol{\sigma})$ the social welfare of $\boldsymbol{\sigma}$. Many different social welfare functions have been proposed in the literature of economic theory. Here, we confine out attention to the two best known: *utilitarian* and *egalitarian* social welfare. These functions are defined as follows:

- The *utilitarian* social welfare function is given by $\mathsf{usw}(\boldsymbol{\sigma}) = \sum_{i \in N} \mathsf{pay}_i(\boldsymbol{\sigma})$.
- The *egalitarian* social welfare function is given by $\mathsf{esw}(\boldsymbol{\sigma}) = \min_{i \in N} \{\mathsf{pay}_i(\boldsymbol{\sigma})\}$.

For simplicity, for a given game $\mathcal{G}$ and a formula φ, by $\mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi) = \{\boldsymbol{\sigma} \in \mathrm{NE} : \pi(\boldsymbol{\sigma}) \models \varphi\}$ we denote the set of Nash equilibria that satisfy φ, that is, that are a solution to the E-Nash problem of $(\mathcal{G}, \varphi)$. For a fixed social welfare function sw on a game $\mathcal{G}$, by:

- $\mathsf{MaxNE}_{\mathsf{sw}}(\mathcal{G}, \varphi) = \max_{\boldsymbol{\sigma} \in \mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)} \{\mathrm{sw}(\vec{\boldsymbol{\sigma}})\}$, and
- $\mathsf{MinNE}_{\mathsf{sw}}(\mathcal{G}, \varphi) = \min_{\boldsymbol{\sigma} \in \mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)} \{\mathrm{sw}(\vec{\boldsymbol{\sigma}})\}$

we denote the maximal and minimal social welfare achieved over a Nash equilibrium profile, respectively, satisfying a given specification φ.

The values of MaxNE and MinNE determine how good or bad the E-Nash solutions are from the perspective of the agents in the game collectively. Here, we consider both the decision and function problem.

Definition 2 (Threshold social welfare) For a given mp game $\mathcal{G}_{\mathsf{mp}}$, a social welfare function sw, and a threshold value t, decide whether there exists a strategy profile $\boldsymbol{\sigma}$ in $\mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)$ such that $t \leq \mathrm{sw}(\boldsymbol{\sigma})$. In case of a positive answer to this decision question, the pair $(\mathcal{G}, \varphi)$ is called *t-increase*.

Analogously, decide whether there exists a strategy profile $\boldsymbol{\sigma}$ in $\mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)$ such that $t \geq \mathrm{sw}(\boldsymbol{\sigma})$. In case of a positive answer to this decision question, the pair $(\mathcal{G}, \varphi)$ is called *t-decrease*.

Definition 3 (Max and Min social welfare) For a given mp game $\mathcal{G}_{\mathsf{mp}}$ and a social welfare function sw, compute $\mathsf{MaxNE}_{\mathsf{sw}}(\mathcal{G}, \varphi)$ and $\mathsf{MinNE}_{\mathsf{sw}}(\mathcal{G}, \varphi)$.

The two definitions above can be instantiated with many different social welfare functions. In the following two subsections, we consider them in the context of the utilitarian and egalitarian welfare measures defined above.

5.1 Social welfare computation with LTL specifications

We first show how to check that a given mp game $\mathcal{G}_{\mathsf{mp}}$ and a LTL specification meets a given threshold t. As the utilitarian and egalitarian functions require different proofs, we address them separately. For the utilitarian function, we have the following.

Theorem 6 For a given mp game $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$, an LTL specification φ, and a threshold value t, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)$ such that $t \leq \mathrm{usw}(\boldsymbol{\sigma})$ is PSPACE-complete. Analogously, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \mathsf{E\text{-}Nash}_{\mathcal{G}}(\varphi)$ such that $t \geq \mathrm{usw}(\boldsymbol{\sigma})$ is PSPACE-complete.

Proof It is enough to show the case $t \leq \mathrm{usw}(\boldsymbol{\sigma})$ as the other one is similar. The solution is a slight modification of the E-Nash problem for mp games with LTL specifications. Consider the arena $A' = \langle \mathrm{N} \cup \{n+1\}, \mathrm{Ac}, \mathrm{St}, s_0, \mathsf{tr}', \lambda \rangle$ with tr' defined as

$$\mathsf{tr}'(\mathsf{a}_1, \ldots, \mathsf{a}_n, \mathsf{a}_{n+1}) = \mathsf{tr}(\mathsf{a}_1, \ldots, \mathsf{a}_n)$$

for every $(\mathsf{a}_1, \ldots, \mathsf{a}_n, \mathsf{a}_{n+1}) \in \mathrm{Ac}^{|\mathrm{N}|+1}$, and the mp game $\mathcal{G}'_{\mathsf{mp}} = \langle A', (\mathsf{w}_i)_{i \in \mathrm{N}}, (\mathsf{w}_{n+1}) \rangle$ with $\mathsf{w}_{n+1}(s) = \sum_{i \in \mathrm{N}} (\mathsf{w}_i(s))$ for every $s \in \mathrm{St}$.

Intuitively, we have included an extra agent in the game, having no effect/impact on the executions, in a way that it carries information about the social welfare of the original game. Indeed, observe that, for every strategy profile $\boldsymbol{\sigma}$ in $\mathcal{G}'_{\mathsf{mp}}$, it holds that

$$\mathsf{pay}'_{n+1}(\vec{\sigma}) = \sum_{i \in \mathrm{N}} \mathsf{pay}'_i(\vec{\sigma}) = \sum_{i \in \mathrm{N}} \mathsf{pay}_i(\vec{\sigma}_{-(n+1)}) = \mathsf{usw}(\vec{\sigma}_{-(n+1)})$$

We can employ the same construction for solving the E-Nash problem for mp games with LTL specifications to solve the threshold problem. It suffices to replace the $\mathsf{LTL}^{\mathsf{Lim}}$ formula $\varphi_{\text{E-Nash}}$ with

$$\varphi^{\mathsf{usw},t}_{\text{E-Nash}} := \varphi_{\text{E-Nash}} \wedge \mathsf{mp}(n+1) \geq t.$$

The computational complexity of the procedure is PSPACE as for E-Nash. The lower bound easily follows from the model checking of LTL. □

For the case of egalitarian social welfare, we have the following.

Theorem 7 For a given mp game $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$, an LTL specification φ, and a threshold value t, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \leq \mathsf{esw}(\boldsymbol{\sigma})$ is PSPACE-complete. Analogously, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \geq \mathsf{esw}(\boldsymbol{\sigma})$ is PSPACE-complete.

Proof It is enough to show the case $t \leq \mathsf{esw}(\boldsymbol{\sigma})$ as the other one is similar. As for the case of utilitarian social welfare functions, the solution is a slight modification of the E-Nash problem for mp games with LTL specifications. Indeed, observe that we can specify that the payoff of agent i is greater than the threshold t by the $\mathsf{LTL}^{\mathsf{Lim}}$ formula $\mathsf{mp}(i) \geq t$. Therefore, specifying that the egalitarian social welfare is at least t can be done by the conjunction $\bigwedge_{i \in \mathrm{N}} \mathsf{mp}(i) \geq t$. Thus, it suffice to replace the $\mathsf{LTL}^{\mathsf{Lim}}$ $\varphi_{\text{E-Nash}}$ for the E-Nash problem with

$$\varphi^{\mathsf{esw},t}_{\text{E-Nash}} := \varphi_{\text{E-Nash}} \wedge \bigwedge_{i \in \mathrm{N}} \mathsf{mp}(i) \geq t.$$

Again, the computational complexity of the procedure is PSPACE and the lower bound follows from the model checking of LTL. □

5.2 Social welfare computation with GR(1) specifications

In this section, we address social welfare threshold problems with GR(1) specifications. The techniques are similar to the ones used in the case of LTL specifications. Firstly, we consider the utilitarian social welfare function. For a given mp game $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$, we build the arena A' and the game $\mathcal{G}'_{\mathsf{mp}}$ analogous to the way it is done in the proof of Theorem 6. Now, to solve the case $t \leq \mathsf{usw}(\boldsymbol{\sigma})$, we adapt the procedure for solving E-Nash for mp games with GR(1) specifications (Theorem 5) as follows. We construct the corresponding multi-weighted graph $W = \langle V, E, (\mathsf{w}'_i)_{i \in \mathrm{N} \cup n+1} \rangle$ where $\mathsf{w}'_{n+1}(v) = \mathsf{w}_{n+1}(s) - t$. Then, solving E-Nash problem for such an instance

corresponds exactly to the threshold social welfare problem $t \leq \text{usw}(\boldsymbol{\sigma})$. For the case $t \geq \text{usw}(\boldsymbol{\sigma})$, we simply define $\mathsf{w}'_{n+1}(v) = t - \mathsf{w}_{n+1}(s)$. To obtain the lower bounds, we reduce from the E-Nash problem for mp games with GR(1) specifications. For the case $t \leq \text{usw}(\boldsymbol{\sigma})$, we set $t = \min\{\mathsf{w}_{n+1}(s) : s \in \text{St}\}$, and the other case, we fix $t = \max\{\mathsf{w}_{n+1}(s) : s \in \text{St}\}$. Thus, we obtain the following result.

Theorem 8 For a given mp game $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \text{N}} \rangle$, a GR(1) specification φ, and a threshold value t, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \leq \text{usw}(\boldsymbol{\sigma})$ is NP-complete. Analogously, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \geq \text{usw}(\boldsymbol{\sigma})$ is NP-complete.

Now we turn our attention to the egalitarian social welfare function. To solve the social threshold problem $t \leq \text{esw}(\boldsymbol{\sigma})$, we directly adapt from the procedure for solving E-Nash for mp games with GR(1) specifications (Theorem 5). For the game $\mathcal{G}[z]$, we build the underlying graph $\langle V, E, (\mathsf{w}'_i)_{i \in \text{N}} \rangle$ where $\mathsf{w}'_i(v) = \mathsf{w}_i(s) - (\max\{z_i, t\})$. Then we define the linear programs $\text{LP}(\psi_l)$ and $\text{LP}(\theta_1, \ldots, \theta_n)$ in the same way. Observe that, one of the two linear programs has a solution if and only if there is a path π satisfying φ such that for every player i, $z_i \leq \text{pay}_i(\pi)$ and $t \leq \text{pay}_i(\pi)$. To obtain the lower bound, again, we reduce from the E-Nash problem for mp games with GR(1) specifications. The reduction simply follows from the fact that by fixing $t = \min\{\mathsf{w}_i(s) : i \in \text{N}, s \in \text{St}\}$, we can encode E-Nash problem into the social threshold problem. The case $t \geq \text{esw}(\boldsymbol{\sigma})$ is similar. Therefore, we obtain the following result.

Theorem 9 For a given mp game $\mathcal{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \text{N}} \rangle$, a GR(1) specification φ, and a threshold value t, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \leq \text{esw}(\boldsymbol{\sigma})$ is NP-complete. Analogously, deciding whether there exists a strategy profile $\boldsymbol{\sigma} \in \text{E-Nash}_{\mathcal{G}}(\varphi)$ such that $t \geq \text{esw}(\boldsymbol{\sigma})$ is NP-complete.

The threshold social welfare calculation can be used to approximate the MaxNE and MinNE values of a game, be it either utilitarian or egalitarian. Note that, for every agent $i \in \text{N}$ and every strategy profile $\boldsymbol{\sigma}$ in the game, it holds that

$$\min(\mathsf{w}_i) = \min_{s \in \text{St}}\{\mathsf{w}_i(s)\} \leq \mathsf{pay}_i(\boldsymbol{\sigma}) \leq \max_{s \in \text{St}}\{\mathsf{w}_i(s)\} = \max(\mathsf{w}_i).$$

This establishes a bound also on the social welfare function, which is given by

$$\sum_{i \in \text{N}} \min(\mathsf{w}_i) \leq \text{MinNE}_{\text{sw}}(\mathcal{G}, \varphi) \leq \text{MaxNE}_{\text{sw}}(\mathcal{G}, \varphi) \leq \sum_{i \in \text{N}} \max(\mathsf{w}_i).$$

Moreover, observe that, for two values $t < t'$, if $(\mathcal{G}, \varphi)$ is t-increase but not t'-increase, then it holds that $t \leq \text{MaxNE}_{\text{sw}}(\mathcal{G}, \varphi) < t'$. Analogously, if $(\mathcal{G}, \varphi)$ is t'-decrease, but not t-decrease, then it holds that $t \leq \text{MinNE}_{\text{sw}}(\mathcal{G}, \varphi) < t'$.

These observations allow to apply a bisection-like method to approximate MaxNE and MinNE. Moreover, note that at each iteration of the method, the absolute error is halved, which ensures linear convergence of the method [30]. Particularly, we obtain an approximation of the values within a fixed tolerance $\epsilon > 0$ in a number n of iterations bounded by $n_\epsilon = \lceil \log_2(\frac{b-a}{\epsilon}) \rceil$, with $a = \sum_{i \in \text{N}} \min(\mathsf{w}_i)$ and $b = \sum_{i \in \text{N}} \max(\mathsf{w}_i)$.

6 Other rational verification problems

E-Nash is, we believe, the most fundamental problem in the rational verification framework, but it is not the only one. The two other key problems are A-Nash and Non-emptiness. The former is the dual problem of E-Nash, which asks, given a game $\mathcal{G}$ and a specification ϕ, whether ϕ is satisfied in *all* Nash equilibria of $\mathcal{G}$. The latter simply asks whether the game $\mathcal{G}$ has at least one Nash equilibrium, and it can be thought of as the special case of E-Nash where the specification ϕ is any tautology.

We can conclude from (the proofs of) the results presented so far, which are summarised in Table 1, that while A-Nash for GR(1) games is also PSPACE and FPT, respectively, in case of LTL and GR(1) specifications, for mp games the problem is, respectively, PSPACE and coNP, in each case. In addition, we can also conclude that whereas Non-emptiness for GR(1) games is FPT, for mp games is NP-complete. These results contrast with those when players' goals are general LTL formulae, where all problems are 2EXPTIME-complete since LTL synthesis, which is 2EXPTIME-hard [28], can be encoded. These results also contrast with those presented in [14], where it is shown that, in succinct model representations given by iterated Boolean games or reactive modules, all problems in the rational verification framework can be polynomially reduced to Non-emptiness, which clearly cannot be the case here, unless the whole polynomial hierarchy collapses.

7 Concluding remarks

We have presented improved complexity results for rational verification problems in three different settings: in the analysis of response properties of reactive systems modelled as multiagent systems; verification of mean-payoff games; and verification of collective properties of multiagent systems through the analysis of social welfare properties. The first scenario mostly concerns the verification of qualitative properties of reactive systems; the second the verification of quantitative properties; and the third the verification of "community" properties, as opposed to individual properties of agents in a system. In the remainder of this article, we discuss further the impact and relevance of our results in these three areas.

Reactive systems The logical analysis of reactive systems is typically carried out using either linear temporal logics, such as LTL, or branching time temporal logics, such as CTL and CTL*. Such analysis may involve verifying that a temporal logic property holds in a given system (model checking) or automatically constructing the system from a temporal logic specification (automated synthesis). Rational verification subsumes both problems, and applies to systems modelled in a distributed way as a collection of semi-autonomous agents (a multiagent system). Despite the greater scope of rational verification with respect to both model checking and automated synthesis, previous work has shown that the overall complexity of rational verification is typically not higher/worse than the combined complexity of the associated synthesis problem. This connection also transfers when considering goals expressed in the GR(1) fragment of LTL, where an initial solution in 2EXPTIME is reduced to complexities lying in the polynomial hierarchy. However, to do so, careful attention must be paid to how the additional game-theoretic analysis that rational verification entails must be done without blowing up the combined computational complexity.

This is particularly important since, in rational verification, strategies for multiple agents must be synthesised, rather than a single model for a reactive system.

Mean-payoff games In the computer science literature, mean-payoff games have been considered as a way of understanding the long-term behaviour (the average performance) of a system—the most common setting is that of a two-player game in which one of the players model the system and the other player models the environment. From a game-theoretic point of view, these are two-player games, which in a perfect information setting can be solved in NP∪coNP, thus without a known polynomial time algorithm to solve them. In case of rational verification with mean-payoff objectives, the problem is definitely harder, (unless P=NP, which is unlikely). We have shown that if the principal has an LTL goal, the problem matches the complexity of LTL model checking, a complexity gap that cannot be avoided since LTL model checking is a particular case. But, even with GR(1) specifications, the problem is very likely to be strictly harder than solving (two-player perfect-information) mean-payoff games since we have shown that with mean-payoff objectives the problem is NP-Complete.

Social welfare While rational verification tends to privilege the preferences of individual agents in a system, social welfare measures focus, instead, on what is considered to be best for a society of agents. Because of this, our results regarding social welfare outcomes may complement nicely the analysis performed in rational verification as originally defined, where the perfromance of society as a whole was irrelevant. We have shown that even in this scenario, better complexity results can be achieved with respect to the complexity of the problem when only individual preferences are considered, as in a Nash equilibrium. In the specific scenario that we considered in the paper, we have shown that the problem is PSPACE-complete, and therefore still efficient with respect to the space complexity of the problem.

Future work A limitation in adopting widely the use of rational verification instead of other reasoning techniques is its combined complexity, which is closely related to the complexity of associated automated synthesis problems. Our results are important because they show that for several significant settings, rational verification can be done with polynomial space algorithms. These results are much more attractive than in the general case, and hold out the hope of efficient practical tools (c.f. the *Equilibrium Verification Environment* (EVE) [20, 21], a tool for the automated analysis of temporal equilibrium properties). Further practical implementations thus seem to be a natural step forward towards the deployment of rational verification in more realistic scenarios.

8 Concluding remarks

We have presented improved complexity results for rational verification problems in three different settings: in the analysis of response properties of reactive systems modelled as multiagent systems; verification of mean-payoff games; and verification of collective properties of multiagent systems through the analysis of social welfare properties. The first scenario mostly concerns the verification of qualitative properties of reactive systems; the second the verification of quantitative properties; and the third the verification of "community"

properties, as opposed to individual properties of agents in a system. In the remainder of this article, we discuss further the impact and relevance of our results in these three areas.

Reactive systems The logical analysis of reactive systems is typically carried out using either linear temporal logics, such as LTL, or branching time temporal logics, such as CTL and CTL*. Such analysis may involve verifying that a temporal logic property holds in a given system (model checking) or automatically constructing the system from a temporal logic specification (automated synthesis). Rational verification subsumes both problems, and applies to systems modelled in a distributed way as a collection of semi-autonomous agents (a multiagent system). Despite the greater scope of rational verification with respect to both model checking and automated synthesis, previous work has shown that the overall complexity of rational verification is typically not higher/worse than the combined complexity of the associated synthesis problem. This connection also transfers when considering goals expressed in the GR(1) fragment of LTL, where an initial solution in 2EXPTIME is reduced to complexities lying in the polynomial hierarchy. However, to do so, careful attention must be paid to how the additional game-theoretic analysis that rational verification entails must be done without blowing up the combined computational complexity. This is particularly important since, in rational verification, strategies for multiple agents must be synthesised, rather than a single model for a reactive system.

Mean-payoff games In the computer science literature, mean-payoff games have been considered as a way of understanding the long-term behaviour (the average performance) of a system—the most common setting is that of a two-player game in which one of the players model the system and the other player models the environment. From a game-theoretic point of view, these are two-player games, which in a perfect information setting can be solved in NP∪coNP, thus without a known polynomial time algorithm to solve them. In case of rational verification with mean-payoff objectives, the problem is definitely harder, (unless P=NP, which is unlikely). We have shown that if the principal has an LTL goal, the problem matches the complexity of LTL model checking, a complexity gap that cannot be avoided since LTL model checking is a particular case. But, even with GR(1) specifications, the problem is very likely to be strictly harder than solving (two-player perfect-information) mean-payoff games since we have shown that with mean-payoff objectives the problem is NP-Complete.

Social welfare While rational verification tends to privilege the preferences of individual agents in a system, social welfare measures focus, instead, on what is considered to be best for a society of agents. Because of this, our results regarding social welfare outcomes may complement nicely the analysis performed in rational verification as originally defined, where the perfromance of society as a whole was irrelevant. We have shown that even in this scenario, better complexity results can be achieved with respect to the complexity of the problem when only individual preferences are considered, as in a Nash equilibrium. In the specific scenario that we considered in the paper, we have shown that the problem is PSPACE-complete, and therefore still efficient with respect to the space complexity of the problem.

Future work A limitation in adopting widely the use of rational verification instead of other reasoning techniques is its combined complexity, which is closely related to the complexity of associated automated synthesis problems. Our results are important because they

show that for several significant settings, rational verification can be done with polynomial space algorithms. These results are much more attractive than in the general case, and hold out the hope of efficient practical tools (c.f. the *Equilibrium Verification Environment* (EVE) [20, 21], a tool for the automated analysis of temporal equilibrium properties). Further practical implementations thus seem to be a natural step forward towards the deployment of rational verification in more realistic scenarios.

Acknowledgements Wooldridge gratefully acknowledges the support of the ERC under Advanced Grant 291528 ("RACE"), and the support of the Alan Turing Institute in London. Najib acknowledges the support of ERC Starting Grant 759969 (AV-SMP). Perelli acknowledges the support of the ERC project "WhiteMech" (grant agreement No 834228) and the EU ICT-48 2020 project TAILOR (No. 952215).

Funding Open Access funding enabled and organized by CAUL and its Member Institutions

Declarations

Conflict of Interests The authors declare that they have no conflict of interest.

References

1. Alur, R., Henzinger, T., Kupferman, O.: Alternating-time temporal logic. J. ACM **49**(5), 672–713 (2002)
2. Alur, R., La Torre, S.: Deterministic generators and games for LTL fragments. ACM Trans. Comput. Log. **5**(1), 1–25 (2004)
3. Bloem, R., Chatterjee, K., Greimel, K., Henzinger, T.A., Jobstmann, B.: Robustness in the presence of liveness. In: Touili, T., Cook, B., Jackson, P.B. (eds.) Computer Aided Verification, 22nd International Conference, CAV 2010, Edinburgh, UK, July 15-19, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6174, pp. 410–424. Springer (2010)
4. Bloem, R., Jobstmann, B., Piterman, N., Pnueli, A., Sa'ar, Y.: Synthesis of reactive(1) designs. J. Comput. Syst. Sci. **78**(3), 911–938 (2012)
5. Boker, U., Chatterjee, K., Henzinger, T., Kupferman, O.: Temporal specifications with accumulative values. ACM Trans. Comput. Logic **15** (4), 27:1–27:25 (2014). https://doi.org/10.1145/2629686
6. Calude, C., Jain, S., Khoussainov, B., Li, W., Stephan, F.: Deciding parity games in quasipolynomial time. In: STOC, pp. 252–263. ACM (2017)
7. Clarke, E., Grumberg, O., Peled, D.: Model Checking. MIT Press (2002)
8. Clarke, E.M., Grumberg, O., Kroening, D., Peled, D., Veith, H.: Model Checking, 2nd edn. MIT Press (2018)
9. Condurache, R., Filiot, E., Gentilini, R., Raskin, J.: The complexity of rational synthesis. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) 43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy, LIPIcs, vol. 55, pp. 121:1–121:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.ICALP.2016.121 (2016)
10. Condurache, R., Oualhadj, Y., Troquard, N.: The complexity of rational synthesis for concurrent games. In: Schewe, S., Zhang, L. (eds.) 29th International Conference on Concurrency Theory,

CONCUR 2018, September 4-7, 2018, Beijing, China, LIPIcs, vol. 118, pp. 38:1–38:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik. https://doi.org/10.4230/LIPIcs.CONCUR.2018.38 (2018)
11. Emerson, E.: Temporal and modal logic. In: Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics, pp. 996–1072. Elsevier (1990)
12. Filiot, E., Gentilini, R., Raskin, J.F.: Rational Synthesis Under Imperfect Information. In: LICS, pp. 422–431. ACM (2018)
13. Fisman, D., Kupferman, O., Lustig, Y.: Rational Synthesis. In: TACAS, LNCS, vol. 6015, pp. 190–204. Springer (2010)
14. Gao, T., Gutierrez, J., Wooldridge, M.: Iterated Boolean Games for Rational Verification. In: AAMAS, pp. 705–713. ACM (2017)
15. Gutierrez, J., Harrenstein, P., Wooldridge, M.: Expresiveness and Complexity Results for Strategic Reasoning. In: CONCUR, LIPIcs, vol. 42, pp. 268–282. Schloss Dagstuhl (2015)
16. Gutierrez, J., Harrenstein, P., Wooldridge, M.: Iterated Boolean Games. Inf. Comput. **242**, 53–79 (2015)
17. Gutierrez, J., Harrenstein, P., Wooldridge, M.: From Model Checking to Equilibrium Checking: Reactive Modules for Rational Verification. Artif. Intell. **248**, 123–157 (2017)
18. Gutierrez, J., Harrenstein, P., Wooldridge, M.: Reasoning about equilibria in game-like concurrent systems. Ann. Pure Appl. Logic **168**(2), 373–403 (2017)
19. Gutierrez, J., Murano, A., Perelli, G., Rubin, S., Wooldridge, M.: Nash equilibria in concurrent games with lexicographic preferences. In: IJCAI, pp. 1067–1073. https://doi.org/10.24963/ijcai.2017/148 (2017)
20. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.: EVE: A tool for temporal equilibrium analysis. In: ATVA, LNCS, vol. 11138, pp. 551–557. Springer (2018)
21. Gutierrez, J., Najib, M., Perelli, G., Wooldridge, M.J.: Automated temporal equilibrium analysis: Verification and synthesis of multi-player games. Artif. Intell. **287**, 103353 (2020). https://doi.org/10.1016/j.artint.2020.103353
22. Gutierrez, J., Perelli, G., Wooldridge, M.: Imperfect information in reactive modules games. Inf. Comput. **261**(Part), 650–675 (2018)
23. Kupferman, O.: Automata Theory and Model Checking Handbook of TCS (2015)
24. Kupferman, O., Perelli, G., Vardi, M.: Synthesis with rational environments. Ann. Math. Artif. Intell. **78**(1), 3–20 (2016)
25. Osborne, M., Rubinstein, A.: A Course in Game Theory. MIT Press (1994)
26. Piterman, N., Pnueli, A.: Faster solutions of Rabin and Streett games. In: LICS, pp. 275–284. https://doi.org/10.1109/LICS.2006.23 (2006)
27. Pnueli, A.: The temporal logic of programs. In: FOCS, pp. 46–57. IEEE (1977)
28. Pnueli, A., Rosner, R.: On the synthesis of a reactive module. In: POPL, pp. 179–190. ACM Press (1989)
29. Rauch Henzinger, M., Telle, J.: Faster algorithms for the nonemptiness of streett automata and for communication protocol pruning. In: SWAT, pp. 16–27 (1996)
30. Sikorski, K: Bisection is optimal. Numer. Math. **40**(1), 111–117 (1982)
31. Steeples, T., Gutierrez, J., Wooldridge, M.J.: Mean-payoff games with ω-regular specifications. In: Dignum, F., Lomuscio, A., Endriss, U., Nowé, A. (eds.) AAMAS '21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, United Kingdom, May 3-7, 2021, pp. 1272–1280. ACM (2021)
32. Ummels, M., Wojtczak, D.: The Complexity of Nash Equilibria in Limit-Average Games. In: CONCUR, pp. 482–496 (2011). https://doi.org/10.1007/978-3-642-23217-6_32
33. Wooldridge, M., Gutierrez, J., Harrenstein, P., Marchioni, E., Perelli, G., Toumi, A.: Rational verification: From model checking to equilibrium checking. In: AAAI, pp. 4184–4191. AAAI Press (2016)
34. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theor. Comput. Sci. **158**(1), 343–359 (1996). https://doi.org/10.1016/0304-3975(95)00188-3