On Computational Tractability for Rational Verification

Julian Gutierrez
Monash University
Melbourne, Australia
julian.gutierrez@monash.edu

Muhammad Najib
Technische Universität Kaiserslautern
Kaiserslautern, Germany
najib@cs.uni-kl.de

Giuseppe Perelli
Sapienza University of Rome
Rome, Italy
perelli@diag.uniroma1.it

Michael Wooldridge
University of Oxford
Oxford, UK
mjw@cs.ox.ac.uk

Rational verification involves checking which temporal logic properties hold of a concurrent and multiagent system, under the assumption that agents in the system choose strategies in a game theoretic equilibrium. Rational verification can be understood as a counterpart of model checking for multiagent systems, but while 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 more intractable: 2EXPTIME-complete with LTL specifications, even when using explicit-state system representations. In this paper 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 most response properties of reactive systems. We also provide improved complexity results for rational verification when considering players' goals given by mean-payoff utility functions—arguably the most widely used quantitative objective for agents in concurrent and multiagent systems. In particular, we show that for a number of relevant settings, rational verification can be done in polynomial space or even in polynomial time.

1 Introduction

The formal verification of systems using temporal logics such as LTL and CTL [7] is a major research area, which has led to the development of an impressive number of industrial-strength verification tools and techniques. Arguably the most successful technique within formal verification is model checking, which can be done in polynomial space for LTL specifications and even in polynomial time for CTL specifications [6]. In the context of multiagent systems, *rational verification* forms a natural counterpart of model checking [12, 28, 13]. 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 (*e.g.*, Nash) equilibrium. Unlike model checking, rational verification is still in its infancy: the main ideas, formal models, and reasoning techniques underlying rational verification are under development, while current tool support is limited and cannot yet handle systems of industrial size [26, 16].

One key difficulty is that rational verification is computationally much harder than model checking, because checking equilibrium properties requires quantifying over the strategies available to players in the system. Rational verification is also different from model checking in the kinds of properties that each technique tries to check: while model checking is interested in correctness with respect to *any* possible behaviour of a system, rational verification is interested only in behaviours that can be *sustained by a Nash equilibrium*, when a multiagent system is modelled as a multi-player game. This, in particular, adds a new ingredient to the verification problem, as it is now necessary to take into account the *preferences* of

Submitted to:
SR 2020

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)

Table 1: Summary of main complexity results.

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. In this case, rational verification with respect to a specification φ is 2EXPTIME-complete, regardless of whether the representation of the system is given succinctly [13, 12] or explicitly simply as a finite-state labelled transition graph [11].

In this paper, we address this issue and provide complexity results that greatly improve on the 2EXPTIME-complete result of the general case. In particular, we consider games where the goals of players are represented as either GR(1) *formulae* (an important fragment of LTL that can express most response properties of a concurrent and reactive system [3]), or *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 represented as concurrent game structures [1].

Our main results, summarised in Table 1, show that in the cases above mentioned, 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. This work has been already published at IJCAI 2019 [17].

Related Work:

Rational verification has been studied for a number of settings, including iterated Boolean games, reactive modules games, and concurrent game structures [12, 13, 11, 14]. In all cases, the problem is 2EXPTIME-complete. Rational verification is also closely related to rational synthesis, which is also 2EXPTIME-complete both in the Boolean case [9] and with rational environments [20]. 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 [18, 8].

Our work also relates to LTL and mean-payoff (mp) games at large. 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 [27] and in NP$\cap$coNP for two-player games [29], and in fact solvable in quasipolynomial time since they can be reduced to two-player perfect-information parity games [5].

2 Preliminaries

Linear Temporal Logic. LTL extends propositional logic with two operators, **X** ("next") and **U** ("until"), for expressing properties of paths [23, 7]. The syntax of LTL is defined with respect to a set AP of atomic propositions as follows:

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

where $p \in \mathrm{AP}$. As usual, we define $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, $\mathbf{F}\varphi \equiv \top\,\mathbf{U}\,\varphi$, and $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$. 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. Formally, the semantics of LTL is given by the following rules:

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

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

General Reactivity of rank 1. The language of *General Reactivity of rank 1*, denoted $\mathsf{GR}(1)$, is the fragment of LTL of formulae written in the following form [3]:

$$(\mathbf{GF}\psi_1 \wedge \ldots \wedge \mathbf{GF}\psi_m) \to (\mathbf{GF}\varphi_1 \wedge \ldots \wedge \mathbf{GF}\varphi_n),$$

where each subformula ψ_i and φ_i is a Boolean combination of atomic propositions.

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

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

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*; $\mathsf{tr} : \mathrm{St} \times \vec{\mathrm{Ac}} \to \mathrm{St}$ is a *transition function* mapping each pair consisting of a state $s \in \mathrm{St}$ and an *action profile* $\vec{\mathsf{a}} \in \vec{\mathrm{Ac}} = \mathrm{Ac}^{\mathrm{N}}$, one for each player, to a successor state; and $\lambda : \mathrm{St} \to 2^{\mathrm{AP}}$ is a labelling function, mapping every state to a subset of *atomic propositions*.

We sometimes call an action profile $\vec{\mathsf{a}} = (\mathsf{a}_1, \ldots, \mathsf{a}_n) \in \vec{\mathrm{Ac}}$ a *decision*, and denote 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 $\vec{\mathsf{a}}$, we let $\vec{\mathsf{a}}_C$ and $\vec{\mathsf{a}}_{-C}$ 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 $\vec{\mathsf{a}}_i$ for $\vec{\mathsf{a}}_{\{i\}}$ and $\vec{\mathsf{a}}_{-i}$ for $\vec{\mathsf{a}}_{\mathrm{N}\setminus\{i\}}$. For two decisions $\vec{\mathsf{a}}$ and $\vec{\mathsf{a}}'$, we write $(\vec{\mathsf{a}}_C, \vec{\mathsf{a}}'_{-C})$ to denote the decision where the actions for players in C are taken from $\vec{\mathsf{a}}$ and the actions for players in $\mathrm{N} \setminus C$ are taken from $\vec{\mathsf{a}}'$.

A *path* $\pi = (s_0, \vec{\mathsf{a}}^0), (s_1, \vec{\mathsf{a}}^1) \cdots$ is an infinite sequence in $(\mathrm{St} \times \vec{\mathrm{Ac}})^{\omega}$ such that $\mathsf{tr}(s_k, \vec{\mathsf{a}}^k) = s_{k+1}$ 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*, $\delta_i : Q_i \times \vec{\mathrm{Ac}} \to Q_i$ 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* $\vec{\sigma} = (\sigma_1, \ldots, \sigma_n)$ is a vector of strategies, one for each player. As with actions, $\vec{\sigma}_i$ denotes the strategy assigned to player i in profile $\vec{\sigma}$. Moreover, by $(\vec{\sigma}_B, \vec{\sigma}'_C)$ we denote the combination of profiles where players in disjoint B and C are assigned their corresponding strategies in $\vec{\sigma}$ and $\vec{\sigma}'$, respectively.

Once a state s and a strategy profile $\vec{\sigma}$ are fixed, the game has an *outcome*, a path in A, which we denote by $\pi(\vec{\sigma}, s)$. Because strategies are deterministic, $\pi(\vec{\sigma}, s)$ is the unique path induced by $\vec{\sigma}$, that is, the sequence $s_0, s_1, s_2, \ldots$ such that

- $s_{k+1} = \mathsf{tr}(s_k, (\tau_1(q_1^k), \ldots, \tau_n(q_n^k)))$, and
- $q_i^{k+1} = \delta_i(q_i^k, (\tau_1(q_1^k), \ldots, \tau_n(q_n^k)))$, for all $k \geq 0$.

Arenas define the dynamic structure of games, but lack a central aspect of a game: preferences, which give games their strategic structure. A *multi-player game* is obtained from an arena A by associating each player with a goal. We consider multi-player games with $\mathsf{GR}(1)$ and mp goals. A multi-player $\mathsf{GR}(1)$ game is a tuple $\mathscr{G}_{\mathsf{GR}(1)} = \langle A, (\gamma_i)_{i \in \mathrm{N}} \rangle$ where A is an arena and γ_i is the $\mathsf{GR}(1)$ goal for player i. A multi-player mp game is a tuple $\mathscr{G}_{\mathsf{mp}} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$, 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 number. When it is clear from the context, we refer to a multi-player $\mathsf{GR}(1)$ or mp game as a *game* and denote it by $\mathscr{G}$. 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 $\mathsf{w}_i(\pi) = \mathsf{w}_i(s_0)\mathsf{w}_i(s_1)\cdots$ of weights is also induced.

For a $\mathsf{GR}(1)$ game and a path π in it, the payoff of a player i is $\mathsf{pay}_i(\pi) = 1$ if $\lambda(\pi) \models \gamma_i$ and $\mathsf{pay}_i(\pi) = 0$ otherwise. Regarding an mp game, the payoff of player i is $\mathsf{pay}_i(\pi) = \mathsf{mp}(\mathsf{w}_i(\pi))$. Moreover, for a $\mathsf{GR}(1)$ game and a path π, by $\mathrm{Win}(\pi) = \{i \in \mathrm{N} : \lambda(\pi) \models \gamma_i\}$ and $\mathrm{Lose}(\pi) = \{j \in \mathrm{N} : \lambda(\pi) \not\models \gamma_j\}$ 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 $\mathrm{Win}(\vec{\sigma}, s) = \mathrm{Win}(\pi(\vec{\sigma}, s))$ and $\mathrm{Lose}(\vec{\sigma}, s) = \mathrm{Lose}(\pi(\vec{\sigma}, s))$, respectively, the set of winners and losers over the path generated by strategy profile $\vec{\sigma}$ when starting the game from s. Furthermore, we simply write $\pi(\vec{\sigma})$ for $\pi(\vec{\sigma}, s_0)$.

Nash equilibrium. Using payoff functions, we can define the concept of Nash equilibrium [21]. For a game $\mathscr{G}$, a strategy profile $\vec{\sigma}$ is a *Nash equilibrium* of $\mathscr{G}$ if, for every player i and strategy $\sigma'_i \in \mathrm{Str}_i$, we have

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

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

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 $\mathscr{G}$ and a formula φ:

$$\text{Does there exist } \vec{\sigma} \in \mathrm{NE}(\mathscr{G}) \text{ such that } \pi(\vec{\sigma}) \models \varphi?$$

This problem can be instantiated in many ways. For instance, in [12], E-NASH was investigated over *iterated Boolean Games* with specifications and players' goals in LTL, and was proved to be $\mathsf{2EXPTIME}$-complete. Iterated Boolean games is a very natural framework, but it is computationally intractable.

Motivated by this computational limitation, in this paper, we study E-NASH for a number of relevant instantiations of the problem, which we show to have better computational complexity. In particular, we study cases where

- Specifications φ are LTL and players' goals are $\mathsf{GR}(1)$;

- Specifications φ are LTL and players have $\mathtt{mp}$ goals;
- Both the specification φ and the goals are $\mathsf{GR}(1)$;
- Specifications φ are $\mathsf{GR}(1)$ and players have $\mathtt{mp}$ 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 $\mathscr{A} = (\Sigma, Q, q^0, \delta, \Omega)$ 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 $\{(E_1, C_1), \dots, (E_n, C_n)\}$ where $E_k \subseteq Q$ and $C_k \subseteq Q$ for all $k \in [1, n]$. A run ρ is accepting in a DSW $\mathscr{A}$ 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 $\mathit{inf}(\rho) \cap E_k = \varnothing$ or $\mathit{inf}(\rho) \cap C_k \neq \varnothing$.

3 Games of General Reactivity of Rank 1

As indicated before, we solve $\mathsf{GR}(1)$ games in two cases: the first one is when the specification formula is expressed in LTL, while the goals are in $\mathsf{GR}(1)$; the second one when the specification formula as well as the goals belong to $\mathsf{GR}(1)$. First, we provide a general result about a characterization of Nash Equilibrium for $\mathsf{GR}(1)$ given in terms of punishments. We first require some notation.

For a $\mathsf{GR}(1)$ game $\mathscr{G}$, player $j \in \mathrm{N}$, and state $s \in \mathrm{St}$, the strategy profile $\vec{\sigma}_{-j}$ is *punishing* for player j in s if $\pi((\vec{\sigma}_{-j}, \sigma'_j), s) \not\models \gamma_j$, 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 $\mathrm{Pun}_j(\mathscr{G})$ the set of punishing states in $\mathscr{G}$. A pair $(s, \vec{\mathsf{a}}) \in \mathrm{St} \times \vec{\mathrm{Ac}}$ is *punishing-secure* for player j, if $\mathsf{tr}(s, (\vec{\mathsf{a}}_{-j}, \mathsf{a}'_j)) \in \mathrm{Pun}_j(\mathscr{G})$ for every action a'_j.

Theorem 1. *In a given* $\mathsf{GR}(1)$ *game* $\mathscr{G}$*, 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* $(s_k, \vec{\mathsf{a}}^k)$ *of the k-th iteration of* π *is punishing-secure for every* $j \in \mathrm{Lose}(\pi)$.

Proof sketch. From left to right, let $\vec{\sigma} \in \mathrm{NE}(\mathscr{G})$ and π be the ultimately periodic path generated by $\vec{\sigma}$. Assume by contradiction that π is not punishing secure for some $j \in \mathrm{N}$, that is, there is $k \in \mathbb{N}$ and action a'_j such that $\mathsf{tr}(s_k, (\vec{\mathsf{a}}_{-j}, \mathsf{a}'_j)^k) \notin \mathrm{Pun}_j(\mathscr{G})$. Thus, j can deviate at s_k and satisfy γ_j, which is a contradiction to $\vec{\sigma}$ being a Nash equilibrium. From right to left, recall that π can be generated by a finite transducer, say T. Moreover, for every losing player j, there is a punishing strategy profile for j in every $s \in \mathrm{Pun}_j(\mathscr{G})$. Combining T with such punishment strategies, we build a profile $\vec{\sigma}$ that follows the actions prescribed by T, until a losing player j deviates. In such a case, $\vec{\sigma}$ would start punishing player j. Observe that $\mathsf{GR}(1)$ objectives are prefix-independent, which is not true for general LTL objectives. That means that the punishment from the k-th iteration takes effect no matter what prefix $\pi_{\leq k}$ has been played so far. Thus, there is no beneficial deviation for j and $\vec{\sigma}$ is a Nash equilibrium. $\square$

At this point, solving E-NASH can be done as follows:

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 $\mathrm{Pun}_j(\mathscr{G})$;
3. Remove from $\mathscr{G}$ the states that are not punishing for players $j \in L$ and the edges (s, s') that are labelled with an action profile $\vec{\mathsf{a}}$ such that $(s, \vec{\mathsf{a}})$ is not punishing-secure for some $j \in L$, thus obtaining a game $\mathscr{G}^{-L}$;

4. Check whether there exists an ultimately periodic path π in $\mathscr{G}^{-L}$ such that $\pi \models \varphi \wedge \bigwedge_{i \in W} \gamma_i$ holds.

The four steps described in the above procedure yield Algorithm 1, which solves the problem at hand.

Algorithm 1: E-NASH of $\mathsf{GR}(1)$ games.

```
1 Input: A game $\mathscr{G}_{\mathsf{GR}(1)}$ and a specification formula $\varphi$.
2 for $i \in \mathrm{N}$ do
3     Compute $\mathrm{Pun}_i(\mathscr{G})$
4 for $W \subseteq \mathrm{N}$ do
5     Compute $\mathscr{G}^{-L}$
6     if $\pi \models (\varphi \wedge \bigwedge_{i \in W} \gamma_i)$ for some $\pi \in \mathscr{G}^{-L}$ then
7         return Accept
8 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 check line 3: this step can be done in polynomial time, as we now show.

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

Proof. We reduce the problem to computing the winning region of a suitably defined Streett game of index $k = 1$, whose complexity is known to be $O(mn^{k+1}kk!)$ [22]. 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 $A' = \langle \mathrm{N}, \mathrm{Ac}, \mathrm{St}', s'_0, \mathsf{tr}' \rangle$ [1] where

- $\mathrm{St}' = \mathrm{St} \times \{0, \ldots, m_j\} \times \{0, \ldots, n_j\}$;
- $s'_0 = (s_0, 0, 0)$;
- $\mathsf{tr}'((s, \iota_1, \iota_2), \vec{\mathsf{a}}) = (\mathsf{tr}(s, \vec{\mathsf{a}}), \iota'_1, \iota'_2)$ where

$$\iota'_1 = \begin{cases} 1, & \text{if } \iota_1 = 0 \\ \iota_1, & \text{if } \iota_1 \neq 0 \text{ and } s \not\models \psi^j_{\iota_1}, \text{ and} \\ (\iota_1 \oplus_{(m_j+1)} 1), & \text{otherwise} \end{cases}$$

$$\iota'_2 = \begin{cases} 1, & \text{if } \iota_2 = 0 \\ \iota_2, & \text{if } \iota_2 \neq 0 \text{ and } s \not\models \theta^j_{\iota_2} \text{ [2]} \\ (\iota_2 \oplus_{(n_j+1)} 1), & \text{otherwise} \end{cases}$$

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

[2] 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 $C_j = \mathrm{St} \times \{0\} \times \{0,\ldots,n_j\}$ and $E_j = \mathrm{St} \times \{0,\ldots,m_j\} \times \{0\}$. 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 of index 1 with (C_j, E_j) being the only Streett pair. As this can be done in polynomial time, we proved the statement. □

Based on Theorem 2, we have the following result.

Corollary 1. *The* E-NASH *problem for* $\mathsf{GR}(1)$ *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 $\mathscr{K}$ can be easily encoded as an instance of E-NASH where $\mathscr{G}$ is played over a Kripke structure $\mathscr{K}$, taken to be its arena, players' goals being tautologies, and the specification being $\neg\varphi$. In such a case, we have that $\mathscr{K} \models \varphi$ if and only if E-NASH for the pair $(\mathscr{G}, \varphi)$ has a negative answer. □

Corollary 1 sharply contrasts with the same result in case the goals of the players are general LTL formulae. In this more general case, E-NASH is $\mathsf{2EXPTIME}$-complete.

The special case of $\mathsf{GR}(1)$ **specifications.** One of the hardest parts of Algorithm 1 is line 6, where an LTL model checking problem has to be solved, making the running time of the whole procedure exponential in the size of the specification and goals of the players. As we show in this section, a way to drastically reduce the complexity of our decision procedure is to let the specification be in $\mathsf{GR}(1)$ too. 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 $\mathsf{GR}(1)$ specification φ can be regarded as a Streett condition of index 1 over an arena A' suitably constructed from the original arena A. Thus, by denoting (C_φ, E_φ) and (C_i, E_i) the Streett pairs corresponding to the $\mathsf{GR}(1)$ 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 $\mathscr{A} = \langle \vec{\mathrm{Ac}}, \mathrm{St}', s'_0, \mathsf{tr}, \Omega \rangle$ 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 $\mathsf{GR}(1)$ 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 [25, 19]. 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 $\mathscr{G}$ and $\varphi, \gamma_1, \ldots, \gamma_N$) plus the complexity of the emptiness problem for a Streett automaton whose size is polynomial in $\mathscr{G}$ $\varphi, \gamma_1, \ldots, \gamma_N$, and exponential in the number of players.

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

Theorem 3. *For a given* $\mathsf{GR}(1)$ *game* $\mathscr{G}$ *and a* $\mathsf{GR}(1)$ *formula* φ*, the* E-NASH *problem can be solved in time that is polynomial in* $|\mathrm{St}|$*,* $|\mathrm{Ac}|$*, and* $|\varphi|$*,* $|\gamma_1|, \ldots, |\gamma_N|$ *and exponential in the number of players* $|\mathrm{N}|$*. 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 [27].

For a player i and a state $s \in \mathrm{St}$, by $\mathtt{pun}_i(s)$ 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 [29]. Thus, it is in $\mathsf{NP} \cap \mathsf{coNP}$, 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 $(s, \vec{\mathsf{a}})$ is z-secure for i if $\mathtt{pun}_i(\mathsf{tr}(s, (\vec{\mathsf{a}}_{-i}, \mathsf{a}'_i))) \leq z$ for every $\mathsf{a}'_i \in \mathrm{Ac}$.

Theorem 4. *For every* mp *game* $\mathscr{G}$ *and ultimately periodic path* $\pi = (s_0, \vec{\mathsf{a}}_0), (s_1, \vec{\mathsf{a}}^1), \ldots$*, the following are equivalent*

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

Proof sketch. $(1) \Rightarrow (2)$: We prove by contraposition. Let z_i be the largest value player i can get by deviating from π, and let $k \in \mathbb{N}$ be such that $z_i = \mathtt{pun}_i(\mathsf{tr}(s_k, (\vec{\mathsf{a}}_{-i}, \mathsf{a}'_i)))$, *i.e.*, i can get as much as z_i by "going alone" at s_k. Suppose further that $\mathsf{pay}_i(\pi) < z_i$. Thus, player i would deviate at s_k since i can get better payoff by not following π—which is a contradiction to π being a path induced by a Nash equilibrium.

$(2) \Rightarrow (1)$: Define strategy profile $\vec{\sigma}$ that follows π as long as no-one has deviated from π. In such a case where player i deviates on the k-th iteration, the strategy profile $\vec{\sigma}_{-i}$ starts playing the z_i-secure strategy for player i that guarantees the payoff of player i to be less than z_i. Therefore, we have $\mathsf{pay}_i(\pi(\vec{\sigma}_{-i}, \sigma'_i)) \leq z_i \leq \mathsf{pay}_i(\pi)$, for every possible strategy σ'_i of player i (the second inequality is due to condition $2(b)$). Thus, there is no beneficial deviation for player i and π is a path induced by 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 $\mathscr{G}$. Similarly to the case of $\mathsf{GR}(1)$ games, we have the following procedure.

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

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 $\mathsf{GR}(1)$ games. Regarding the last step, its complexity depends on the specification language. For the case of φ being an LTL formula, consider the formula

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

written in the language $\mathsf{LTL}^{\mathsf{Lim}}$, an extension of LTL where statements about mean-payoff values over a given weighted arena can be made [4]. 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 viceversa. 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 [4], 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 $\mathsf{GR}(1)$ games, we can summarize the procedure in the following algorithm (Algorithm 2).

Algorithm 2: E-NASH of mp games.

```
1 Input: A game $\mathcal{G}_{\mathsf{mp}}$ and a specification formula $\varphi$.
2 for $i \in \mathrm{N}$ and $s \in \mathrm{St}$ do
3     Compute $\mathrm{pun}_i(\mathcal{G})$
4 for $\vec{z} \in \{\mathrm{pun}_i(s) : s \in \mathrm{St}\}^{\mathrm{N}}$ do
5     Compute $\mathcal{G}[z]$
6     if $\pi \models \varphi_{\text{E-NASH}}$ for some $\pi \in \mathcal{G}[z]$ then
7         return Accept
8 return Reject
```

The special case of $\mathsf{GR}(1)$ specifications. As in the case of $\mathsf{GR}(1)$ games, here we show that restricting the specification language to $\mathsf{GR}(1)$ lowers the complexity also for mp games. The reason is that the path finding problem for $\mathsf{GR}(1)$ specifications can be done while avoiding model-checking of an $\mathsf{LTL}^{\mathsf{Lim}}$ formula. In order to do this, we follow a different approach. Using an mp game $\mathcal{G}$ and a $\mathsf{GR}(1)$ specification φ we define a linear program such that the linear program has a solution if and only if the pair $(\mathcal{G},\varphi)$ is an instance of E-NASH. In particular, this approach is similar to the technique used in [15, 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* $\mathsf{GR}(1)$ *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 $\mathsf{GR}(1)$ specification.

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

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

and let $V(\psi_l)$ and $V(\theta_r)$ be the subsets 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 $\langle V, E\rangle$ be the underlying graph, and for every edge $e \in E$ 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 $\mathsf{src}(e) = \{v \in V : \exists w\, e = (v,w) \in E\}$; $\mathsf{trg}(e) = \{v \in V : \exists w\, e = (w,v) \in E\}$; $\mathsf{out}(v) = \{e \in E : \mathsf{src}(e) = v\}$; and $\mathsf{in}(v) = \{e \in E : \mathsf{trg}(e) = v\}$.

Consider ψ_l for some $1 \leq l \leq m$, and define the linear program $\mathsf{LP}(\psi_l)$ 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 $a \in \mathrm{N}$, $\Sigma_{e \in E} \mathsf{w}_a(\mathsf{src}(e)) x_e \geq 0$ — this enforces that the total sum of any solution is positive;

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

Eq5: for each $v \in V$, $\Sigma_{e \in \mathsf{out}(v)} x_e = \Sigma_{e \in \mathsf{in}(v)} x_e$ — this condition 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 $\mathsf{LP}(\psi_l)$ admits a solution if and only if there exists a path π in $\mathcal{G}$ such that $z_i \leq \mathsf{pay}_i(\pi)$ for every player i and visits $V(\psi_l)$ only *finitely many times*. In addition, consider the linear program $\mathsf{LP}(\theta_1, \dots, \theta_n)$ 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 $a \in \mathrm{N}$, $\Sigma_{e \in E} \mathsf{w}_a(\mathsf{src}(e)) x_e \geq 0$ — this enforces that the total sum of any solution is positive;

Eq4: for all $1 \leq r \leq n$, $\Sigma_{\mathsf{src}(e) \cap V(\theta_r) \neq \emptyset} x_e \geq 1$ — this ensures that for every $V(\theta_r)$ at least one state is in the cycle;

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

In this case, $\mathsf{LP}(\theta_1, \dots, \theta_n)$ admits a solution if and only if there exists a path π such that $z_i \leq \mathsf{pay}_i(\pi)$ 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 $z_i \leq \mathsf{pay}_i(\pi)$ 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 [27] and observe that if φ is true, then the problem is equivalent to checking whether the mp game has a Nash equilibrium. □

5 Other Rational Verification Problems

E-NASH is, arguably, 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 is 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 [24], can be encoded. These results also contrast with those presented in [10], 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 reduced to NON-EMPTINESS, which clearly cannot be the case here, unless the whole polynomial hierarchy collapses.

Acknowledgements. This work was carried out while Najib and Gutierrez were at the University of Oxford and Perelli at the University of Leicester. Najib was supported by the Indonesia Endowment Fund for Education (LPDP), and by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759969).d Perelli received financial support from the European Research Council under the European Union's Horizon 2020 Programme through the ERC Advanced Investigator Grant 834228 ("WhiteMech"). Wooldridge was supported by a JPMorgan faculty grant, and by the Alan Turing Institute. An extended version of this paper can be found in [17].

References

[1] Rajeev Alur, Thomas Henzinger & Orna Kupferman (2002): *Alternating-Time Temporal Logic*. Journal of the ACM 49(5), pp. 672–713.

[2] Rajeev Alur & Salvatore La Torre (2004): *Deterministic Generators and Games for LTL Fragments*. ACM Transactions on Computational Logic 5(1), pp. 1–25.

[3] Roderick Bloem, Barbara Jobstmann, Nir Piterman, Amir Pnueli & Yaniv Sa'ar (2012): *Synthesis of Reactive(1) designs*. Journal of Computer and System Sciences 78(3), pp. 911–938.

[4] Udi Boker, Krishnendu Chatterjee, Thomas Henzinger & Orna Kupferman (2014): *Temporal Specifications with Accumulative Values*. ACM Transactions on Computational Logic 15(4), pp. 27:1–27:25, doi:10.1145/2629686.

[5] Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li & Frank Stephan (2017): *Deciding Parity Games in Quasipolynomial Time*. In: STOC, ACM, pp. 252–263.

[6] Edmund M. Clarke, Orna Grumberg, Daniel Kroening, Doron Peled & Helmut Veith (2018): *Model Checking (2nd edition)*. MIT Press.

[7] E. Allen Emerson (1990): *Temporal and Modal Logic*. In: Handbook of Theoretical Computer Science Volume B: Formal Models and Semantics, Elsevier, pp. 996–1072.

[8] Emmanuel Filiot, Raffaella Gentilini & Jean-François Raskin (2018): *Rational Synthesis Under Imperfect Information*. In: LICS, ACM, pp. 422–431.

[9] Dana Fisman, Orna Kupferman & Yoad Lustig (2010): *Rational Synthesis*. In: TACAS, LNCS 6015, Springer, pp. 190–204.

[10] Tong Gao, Julian Gutierrez & Michael Wooldridge (2017): *Iterated Boolean Games for Rational Verification*. In: AAMAS, ACM, pp. 705–713.

[11] Julian Gutierrez, Paul Harrenstein & Michael Wooldridge (2015): *Expresiveness and Complexity Results for Strategic Reasoning*. In: CONCUR, LIPIcs 42, Schloss Dagstuhl, pp. 268–282.

[12] Julian Gutierrez, Paul Harrenstein & Michael Wooldridge (2015): *Iterated Boolean Games*. Information and Computation 242, pp. 53–79.

[13] Julian Gutierrez, Paul Harrenstein & Michael Wooldridge (2017): *From Model Checking to Equilibrium Checking: Reactive Modules for Rational Verification*. Artificial Intelligence 248, pp. 123–157.

[14] Julian Gutierrez, Paul Harrenstein & Michael Wooldridge (2017): *Reasoning about Equilibria in Game-like Concurrent Systems*. Annals of Pure and Applied Logic 168(2), pp. 373–403.

[15] Julian Gutierrez, Aniello Murano, Giuseppe Perelli, Sasha Rubin & Michael Wooldridge (2017): *Nash Equilibria in Concurrent Games with Lexicographic Preferences*. In: IJCAI, pp. 1067–1073, doi:10.24963/ijcai.2017/148.

[16] Julian Gutierrez, Muhammad Najib, Giuseppe Perelli & Michael Wooldridge (2018): *EVE: A Tool for Temporal Equilibrium Analysis*. In: ATVA, LNCS 11138, Springer, pp. 551–557.

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

[18] Julian Gutierrez, Giuseppe Perelli & Michael Wooldridge (2018): *Imperfect Information in Reactive Modules games*. Information and Computation 261(Part), pp. 650–675.

[19] Orna Kupferman (2015): *Automata Theory and Model Checking*. Handbook of TCS.

[20] Orna Kupferman, Giuseppe Perelli & Moshe Vardi (2016): *Synthesis with Rational Environments*. Annals of Mathematics and Artificial Intelligence 78(1), pp. 3–20.

[21] Martin J. Osborne & Ariel Rubinstein (1994): *A Course in Game Theory.* MIT Press.

[22] Nir Piterman & Amir Pnueli (2006): *Faster Solutions of Rabin and Streett Games*. In: LICS, pp. 275–284, doi:10.1109/LICS.2006.23.

[23] Amir Pnueli (1977): *The Temporal Logic of Programs*. In: FOCS, IEEE, pp. 46–57.

[24] Amir Pnueli & Roni Rosner (1989): *On the Synthesis of a Reactive Module*. In: POPL, ACM Press, pp. 179–190.

[25] Monika Rauch Henzinger & Jan Telle (1996): *Faster Algorithms for the Nonemptiness of Streett Automata and for Communication Protocol Pruning*. In: SWAT, pp. 16–27.

[26] Alexis Toumi, Julian Gutierrez & Michael Wooldridge (2015): *A Tool for the Automated Verification of Nash Equilibria in Concurrent Games*. In: ICTAC, LNCS 9399, Springer, pp. 583–594.

[27] Michael Ummels & Dominik Wojtczak (2011): *The Complexity of Nash Equilibria in Limit-Average Games*. In: CONCUR, pp. 482–496, doi:10.1007/978-3-642-23217-6-32.

[28] Michael Wooldridge, Julian Gutierrez, Paul Harrenstein, Enrico Marchioni, Giuseppe Perelli & Alexis Toumi (2016): *Rational Verification: From Model Checking to Equilibrium Checking*. In: AAAI, AAAI Press, pp. 4184–4191.

[29] Uri Zwick & Mike Paterson (1996): *The Complexity of Mean Payoff Games on Graphs*. Theoretical Computer Science 158(1), pp. 343 – 359, doi:https://doi.org/10.1016/0304-3975(95)00188-3.