Equilibrium Design for Concurrent Games

Julian Gutierrez
Department of Computer Science, University of Oxford
julian.gutierrez@cs.ox.ac.uk

Muhammad Najib
Department of Computer Science, University of Oxford
mnajib@cs.ox.ac.uk

Giuseppe Perelli
Department of Computer Science, University of Göteborg
giuseppe.perelli@gu.se

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

Abstract

In game theory, *mechanism design* is concerned with the design of incentives so that a desired outcome of the game can be achieved. In this paper, we study the design of incentives so that a desirable equilibrium is obtained, for instance, an equilibrium satisfying a given temporal logic property—a problem that we call *equilibrium design*. We base our study on a framework where system specifications are represented as temporal logic formulae, games as quantitative concurrent game structures, and players' goals as mean-payoff objectives. In particular, we consider system specifications given by LTL and GR(1) formulae, and show that implementing a mechanism to ensure that a given temporal logic property is satisfied on some/every Nash equilibrium of the game, whenever such a mechanism exists, can be done in PSPACE for LTL properties and in NP/Σ_2^{P} for GR(1) specifications. We also study the complexity of various related decision and optimisation problems, such as optimality and uniqueness of solutions, and show that the complexities of all such problems lie within the polynomial hierarchy. As an application, equilibrium design can be used as an alternative solution to the rational synthesis and verification problems for concurrent games with mean-payoff objectives whenever no solution exists, or as a technique to repair, whenever possible, concurrent games with undesirable rational outcomes (Nash equilibria) in an optimal way.

2012 ACM Subject Classification Theory of computation → Modal and temporal logics; Computing methodologies → Multi-agent systems; Theory of computation → Algorithmic game theory

Keywords and phrases Games, Temporal logic, Synthesis, Model checking, Nash equilibrium.

Digital Object Identifier 10.4230/LIPIcs.CONCUR.2019.22

Acknowledgements Najib acknowledges the financial support of the Indonesia Endowment Fund for Education (LPDP), and Perelli the support of the project "dSynMA", funded by the ERC under the European Union's Horizon 2020 research and innovation programme (grant agreement No 772459).

1 Introduction

Over the past decade, there has been increasing interest in the use of game-theoretic equilibrium concepts such as Nash equilibrium in the analysis of concurrent and multi-agent systems (see, *e.g.*, [3, 4, 8, 14, 16, 18, 24]). This work views a concurrent system as a game, with system components (agents) corresponding to players in the game, which are assumed to be acting rationally in pursuit of their individual preferences. Preferences may be specified by associating with each player a temporal logic goal formula, which the player desires to see satisfied, or by assuming that players receive rewards in each state the system visits, and seek to maximise the average reward they receive (the *mean payoff*). A further possibility is to combine goals and rewards: players primarily seek the satisfaction of their goal, and only secondarily seek to maximise their mean payoff. The key decision problems in such settings relate to what temporal logic properties hold on computations of the system that may be generated by players choosing strategies that form a game-theoretic (Nash) equilibrium. These problems are typically computationally complex, since they subsume temporal logic synthesis [34]. If players have LTL goals, for example, then checking whether an LTL formula holds on some Nash equilibrium path in a concurrent game is 2EXPTIME-complete [14, 17, 18], rather than only PSPACE-complete as it is the case for model checking, certainly a computational barrier for the practical analysis and automated verification of reactive, concurrent, and multi-agent systems modelled as multi-player games.

Within this game-theoretic reasoning framework, a key issue is that individually rational choices can cause outcomes that are highly undesirable, and concurrent games also fall prey to this problem. This has motivated the development of techniques for modifying games, in order to avoid bad equilibria, or to facilitate good equilibria. *Mechanism design* is the problem of designing a game such that, if players behave rationally, then a desired outcome will be obtained [27]. Taxation and subsidy schemes are probably the most important class of techniques used in mechanism design. They work by levying taxes on certain actions (or providing subsidies), thereby incentivising players away from some outcomes towards others. The present paper studies the design of subsidy schemes (incentives) for concurrent games, so that a desired outcome (a Nash equilibrium in the game) can be obtained—a problem that we call *Equilibrium design.* We model agents as synchronously executing concurrent processes, with each agent receiving an integer payoff for every state the overall system visits; the overall payoff an agent receives over an infinite computation path is then defined to be the mean payoff over this path. While agents (naturally) seek to maximise their individual mean payoff, the designer of the subsidy scheme wishes to see some temporal logic formula satisfied, either on some or on every Nash equilibrium of the game.

With this model, we assume that the designer – an external principal – has a finite budget that is available for making subsidies, and this budget can be allocated across agent/state pairs. By allocating this budget appropriately, the principal can incentivise players away from some states and towards others. Since the principal has some temporal logic goal formula, it desires to allocate subsidies so that players are rationally incentivised to choose strategies so that the principal's temporal logic goal formula is satisfied in the path that would result from executing the strategies. For this general problem, following [25], we identify two variants of the principal's mechanism design problem, which we refer to as WEAK IMPLEMENTATION and STRONG IMPLEMENTATION. In the WEAK variant, we ask whether the principal can allocate the budget so that the goal is achieved on *some* computation path that would be generated by Nash equilibrium strategies in the resulting system; in the STRONG variation, we ask whether the principal can allocate the budget so that the resulting system has at least one Nash equilibrium, and moreover the temporal logic goal is satisfied on *all* paths that could be generated by Nash equilibrium strategies. For these two problems, we consider goals specified by LTL formulae or GR(1) formulae [5], give algorithms for each case, and classify the complexity of the problem. While LTL is a natural language for the specification of properties of concurrent and multi-agent systems, GR(1) is an LTL fragment that can be used to easily express several prefix-independent properties of computation paths of reactive systems, such as ω-regular properties often used in automated formal verification. We then go on to study variations of these two problems, for example considering *optimality* and *uniqueness* of solutions, and show that the complexities of all such problems lie within the polynomial hierarchy, thus making them potentially amenable to efficient practical implementations.

Table 1 summarises the main computational complexity results in the paper.

	LTL Spec.	GR(1) Spec.
Weak Implementation	PSPACE-complete (Thm. 6)	NP-complete (Thm. 7)
Strong Implementation	PSPACE-complete (Cor. 9)	Σ_2^{P}-complete (Thm. 10)
Opt-WI	FPSPACE-complete (Thm. 14)	$\mathsf{FP}^{\mathsf{NP}}$-complete (Thm. 16)
Opt-SI	FPSPACE-complete (Thm. 22)	$\mathsf{FP}^{\Sigma_2^{\mathsf{P}}}$-complete (Thm. 25)
Exact-WI	PSPACE-complete (Cor. 15)	D^{P}-complete (Cor. 17)
Exact-SI	PSPACE-complete (Cor. 23)	$\mathsf{D}_2^{\mathsf{P}}$-complete (Cor. 26)
UOpt-WI	PSPACE-complete (Cor. 18)	Δ_2^{P}-complete (Cor. 19)
UOpt-SI	PSPACE-complete (Cor. 27)	Δ_3^{P}-complete (Cor. 28)

Table 1 Summary of main complexity results.

2 Preliminaries

Linear Temporal Logic. LTL [33] extends classical propositional logic with two operators, **X** ("next") and **U** ("until"), that can be used to express properties of paths. 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 commonly found in the LTL literature, we use of the following abbreviations: $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$, $\varphi_1 \to \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$, $\mathbf{F}\varphi \equiv \top\,\mathbf{U}\,\varphi$, and $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$.

We interpret formulae of LTL with respect to pairs (α, t), where $\alpha \in (2^{\mathrm{AP}})^\omega$ is an infinite sequence of atomic proposition evaluations 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 formulae 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 GR(1), is the fragment of LTL given by formulae written in the following form [5]:

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

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

Mean-Payoff. For a sequence $r \in \mathbb{R}^\omega$, let $\mathsf{mp}(r)$ be the *mean-payoff* value of r, that is,

$$\mathsf{mp}(r) = \lim \inf_{n\to\infty} \mathsf{avg}_n(r)$$

where, for $n \in \mathbb{N} \setminus \{0\}$, we define $\mathsf{avg}_n(r) = \frac{1}{n}\sum_{j=0}^{n-1} r_j$, with r_j the $(j+1)$th element of r.

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; if needed, we write $\mathrm{Ac}_i(s)$, to denote the set of actions available to player i at s; $s_0 \in \mathrm{St}$ 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 $\vec{\mathsf{a}} = (\mathsf{a}_1, \dots, \mathsf{a}_n) \in \vec{\mathrm{Ac}}$ 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*. Let Str_i be the set of strategies for player i. Note that this definition implies that strategies have *perfect information*[1] and finite memory (although we impose no bounds on memory size).

A *strategy profile* $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$ 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 profile $\vec{\sigma}$ are fixed, the game has an *outcome*, a path in A, denoted 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 $s_0, s_1, s_2, \dots$ such that

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

Furthermore, we simply write $\pi(\vec{\sigma})$ for $\pi(\vec{\sigma}, s_0)$.

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 mp goals. A multi-player mp game is a tuple $\mathcal{G} = \langle A, (\mathsf{w}_i)_{i\in\mathrm{N}}\rangle$, where A is an arena and $\mathsf{w}_i : \mathrm{St} \to \mathbb{Z}$ is a function mapping, for every player i, every state of the arena into an integer number. 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. Unless stated otherwise, for a game $\mathcal{G}$ and a path π in it, the payoff of player i is $\mathsf{pay}_i(\pi) = \mathsf{mp}(\mathsf{w}_i(\pi))$.

Nash equilibrium. Using payoff functions, we can define the game-theoretic concept of

[1] Mean-payoff games with imperfect information are generally undecidable [13].

Nash equilibrium [27]. For a multi-player game $\mathcal{G}$, a strategy profile $\vec{\sigma}$ is a *Nash equilibrium* of $\mathcal{G}$ if, for every player i and strategy σ'_i for player i, we have

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

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

3 From Mechanism Design to Equilibrium Design

We now describe the two main problems that are our focus of study. As discussed in the introduction, such problems are closely related to the well-known problem of *mechanism design* in game theory. Consider a system populated by agents N, where each agent $i \in \mathrm{N}$ wants to maximise its payoff $\mathsf{pay}_i(\cdot)$. As in a mechanism design problem, we assume there is an external *principal* who has a goal φ that it wants the system to satisfy, and to this end, wants to incentivise the agents to act collectively and rationally so as to bring about φ. In our model, incentives are given by *subsidy schemes* and goals by temporal logic formulae.

Subsidy Schemes: A subsidy scheme defines additional imposed rewards over those given by the weight function w. While the weight function w is fixed for any given game, the principal is assumed to be at liberty to define a subsidy scheme as they see fit. Since agents will seek to maximise their overall rewards, the principal can incentivise agents away from performing visiting some states and towards visiting others; if the principal designs the subsidy scheme correctly, the agents are incentivised to choose a strategy profile $\vec{\sigma}$ such that $\pi(\vec{\sigma}) \models \varphi$. Formally, we model a subsidy scheme as a function $\kappa : \mathrm{N} \to \mathrm{St} \to \mathbb{N}$, where the intended interpretation is that $\kappa(i)(s)$ is the subsidy in the form of a natural number $k \in \mathbb{N}$ that would be imposed on player i if such a player visits state $s \in \mathrm{St}$. For instance, if we have $\mathsf{w}_i(s) = 1$ and $\kappa(i)(s) = 2$, then player i gets $1 + 2 = 3$ for visiting such a state. For simplicity, hereafter we write $\kappa_i(s)$ instead of $\kappa(i)(s)$ for the subsidy for player i.

Notice that having an unlimited fund for a subsidy scheme would make some problems trivial, as the principal can always incentivise players to satisfy φ (provided that there is a path in A satisfying φ). A natural and more interesting setting is that the principal is given a constraint in the form of *budget* $\beta \in \mathbb{N}$. The principal then can only spend within the budget limit. To make this clearer, we first define the *cost* of a subsidy scheme κ as follows.

▶ **Definition 1.** *Given a game $\mathcal{G}$ and subsidy scheme κ, we let* $\mathsf{cost}(\kappa) = \sum_{i \in \mathrm{N}} \sum_{s \in \mathrm{St}} \kappa_i(s)$.

We say that a subsidy scheme κ is *admissible* if it does not exceed the budget β, that is, if $\mathsf{cost}(\kappa) \leq \beta$. Let $\mathcal{K}(\mathcal{G}, \beta)$ denote the set of admissible subsidy schemes over $\mathcal{G}$ given budget $\beta \in \mathbb{N}$. Thus we know that for each $\kappa \in \mathcal{K}(\mathcal{G}, \beta)$ we have $\mathsf{cost}(\kappa) \leq \beta$. We write $(\mathcal{G}, \kappa)$ to denote the resulting game after the application of subsidy scheme κ on game $\mathcal{G}$. Formally, we define the application of some subsidy scheme on a game as follows.

▶ **Definition 2.** *Given a game $\mathcal{G} = \langle A, (\mathsf{w}_i)_{i \in \mathrm{N}} \rangle$ and an admissible subsidy scheme κ, we define $(\mathcal{G}, \kappa) = \langle A, (\mathsf{w}'_i)_{i \in \mathrm{N}} \rangle$, where $\mathsf{w}'_i(s) = \mathsf{w}_i(s) + \kappa_i(s)$, for each $i \in \mathrm{N}$ and $s \in \mathrm{St}$.*

We now come to the main question(s) that we consider in the remainder of the paper. We ask whether the principal can find a subsidy scheme that will incentivise players to collectively choose a rational outcome (a Nash equilibrium) that satisfies its temporal logic goal φ. We call this problem *equilibrium design*. Following [25], we define two variants of this problem, a *weak* and a *strong* implementation of the equilibrium design problem. The formal definition of the problems and the analysis of their respective computational complexity are presented in the next sections.

4 Equilibrium Design: Weak Implementation

In this section, we study the weak implementation of the equilibrium design problem, a logic-based computational variant of the principal's mechanism design problem in game theory. We assume that the principal has full knowledge of the game $\mathcal{G}$ under consideration, that is, the principal uses all the information available of $\mathcal{G}$ to find the appropriate subsidy scheme, if such a scheme exists. We now formally define the weak variant of the implementation problem, and study its respective computational complexity, first with respect to goals (specifications) given by LTL formulae and then with respect to GR(1) formulae.

Let $\mathrm{WI}(\mathcal{G}, \varphi, \beta)$ denote the set of subsidy schemes over $\mathcal{G}$ given budget β that satisfy a formula φ in at least one path π generated by $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$. Formally

$$\mathrm{WI}(\mathcal{G}, \varphi, \beta) = \{\kappa \in \mathcal{K}(\mathcal{G}, \beta) : \exists \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ s.t. } \pi(\vec{\sigma}) \models \varphi\}.$$

▶ **Definition 3** (Weak Implementation). *Given a game $\mathcal{G}$, formula φ, and budget β:*

Is it the case that $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$?

In order to solve Weak Implementation, we first characterise the Nash equilibria of a multi-player concurrent game in terms of punishment strategies. To do this in our setting, we recall the notion of secure values for mean-payoff games [36].

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 [38]. Thus, it is in NP ∩ coNP, and note that both player i and coalition $\mathrm{N} \setminus \{i\}$ can achieve the optimal value of the game using *memoryless* strategies. Then, for a player i and a value $z \in \mathbb{R}$, a pair $(s, \vec{\mathsf{a}})$ is z-secure for player i if $\mathtt{pun}_i(\mathsf{tr}(s, (\vec{\mathsf{a}}_{-i}, \mathsf{a}'_i))) \leq z$ for every $\mathsf{a}'_i \in \mathrm{Ac}$. Write $\mathtt{pun}_i(\mathcal{G})$ for the set of punishment values for player i in $\mathcal{G}$.

▶ **Theorem 4.** *For every* mp *game $\mathcal{G}$ 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 $\vec{\sigma} \in \mathrm{NE}(\mathcal{G})$ such that $\pi = \pi(\vec{\sigma}, s_0)$;*
2. *There exists $z \in \mathbb{R}^{\mathrm{N}}$, where $z_i \in \mathtt{pun}_i(\mathcal{G})$ 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. *$z_i \leq \mathsf{pay}_i(\pi)$.*

The characterisation of Nash Equilibria provided in Theorem 4 will allow us to turn the Weak Implementation problem into a *path finding* problem over $(\mathcal{G}, \kappa)$. On the other hand, with respect to the budget β that the principal has at its disposal, the definition of subsidy scheme function κ implies that the size of $\mathcal{K}(\mathcal{G}, \beta)$ is bounded, and particularly, it is bounded by β and the number of agents and states in the game $\mathcal{G}$, in the following way.

▶ **Proposition 5.** *Given a game $\mathcal{G}$ with $|N|$ players and $|\mathrm{St}|$ states and budget β, it holds that*

$$|\mathcal{K}(\mathcal{G}, \beta)| = \frac{\beta + 1}{m} \binom{\beta + m}{\beta + 1},$$

with $m = |N \times \mathrm{St}|$ being the number of pairs of possible agents and states.

From Proposition 5 we derive that the number of possible subsidy schemes is *polynomial* in the budget β and singly *exponential* in both the number of agents and states in the game. At this point, solving Weak Implementation can be done with the following procedure:

1. Guess:
   - a subsidy scheme $\kappa \in \mathcal{K}(\mathcal{G}, \beta)$,
   - a state $s \in \mathrm{St}$ for every player $i \in \mathrm{N}$, and
   - punishment memoryless strategies $(\vec{\sigma}_{-1}, \ldots, \vec{\sigma}_{-n})$ for all players $i \in \mathrm{N}$;
2. Compute $(\mathcal{G}, \kappa)$;
3. Compute $z \in \mathbb{R}^{\mathrm{N}}$;
4. Compute the game $(\mathcal{G}, \kappa)[z]$ by removing the states s such that $\mathtt{pun}_i(s) \leq z_i$ for some player i and the transitions $(s, \vec{\mathsf{a}}_{-i})$ that are not z_i secure for player i;
5. Check whether there exists an ultimately periodic path π in $(\mathcal{G}, \kappa)[z]$ such that $\pi \models \varphi$ and $z_i \leq \mathsf{pay}_i(\pi)$ for every player $i \in \mathrm{N}$.

Since the set $\mathcal{K}(\mathcal{G}, \beta)$ is finitely bounded (Proposition 5), and punishment strategies only need to be memoryless, thus also finitely bounded, clearly step 1 can be guessed nondeterministically. Moreover, each of the guessed elements is of polynomial size, thus this step can be done (deterministically) in polynomial space. Step 2 clearly can be done in polynomial time. Step 3 can also be done in polynomial time since, given $(\vec{\sigma}_{-1}, \ldots, \vec{\sigma}_{-n})$, we can compute z solving $|\mathrm{N}|$ one-player mean-payoff games, one for each player i [38, Thm. 6]. For step 5, we will use Theorem 4 and consider two cases, one for LTL specifications and one for GR(1) specifications. Firstly, for LTL specifications, consider the formula

$$\varphi_{\mathrm{WI}} := \varphi \wedge \bigwedge_{i \in \mathrm{N}} (\mathsf{mp}(i) \geq z_i)$$

written in $\mathsf{LTL}^{\mathsf{Lim}}$ [7], an extension of LTL where statements about mean-payoff values over a given weighted arena can be made.[2] The semantics of the temporal operators of $\mathsf{LTL}^{\mathsf{Lim}}$ is just like the one for LTL over infinite computation paths $\pi = s_0, s_1, s_3 \ldots$. On the other hand, the meaning of $\mathsf{mp}(i) \geq z_i$ is simply that such an atomic formula is true if, and only if, the mean-payoff value of π with respect to player i is greater or equal to z_i, a constant real value; that is, $\mathsf{mp}(i) \geq z_i$ is true in π if and only if $\mathsf{pay}_i(\pi) = \mathsf{mp}(\mathsf{w}_i(\pi))$ is greater or equal than constant value z_i. Formula φ_{WI} corresponds exactly to $2(b)$ in Theorem 4. Furthermore, since every path in $(\mathcal{G}, \kappa)[z]$ satisfies condition $2(a)$ of Theorem 4, every computation path of $(\mathcal{G}, \kappa)[z]$ that satisfies φ_{WI} is a witness to the Weak Implementation problem.

▶ **Theorem 6.** Weak Implementation *with* LTL *specifications is* PSPACE*-complete.*

Proof. Membership follows from the procedure above and the fact that model checking for $\mathsf{LTL}^{\mathsf{Lim}}$ is PSPACE-complete [7]. Hardness follows from the fact that LTL model checking is a special case of Weak Implementation. For instance, consider the case in which all weights for all players are set to the same value, say 0, and the principal has budget $\beta = 0$. ◀

Case with GR(1) specifications. One of the main bottlenecks of our procedure to solve Weak Implementation lies in step 5, where we solve an $\mathsf{LTL}^{\mathsf{Lim}}$ model checking problem. To reduce the complexity of our decision procedure, we consider Weak Implementation with the specification φ expressed in the GR(1) sublanguage of LTL. With this specification language, the path finding problem can be solved without model-checking the $\mathsf{LTL}^{\mathsf{Lim}}$ formula given before. In order to do this, we can define a linear program (LP) such that the LP has a solution if and only if $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$. From our previous procedure, observe that

[2] The formal semantics of $\mathsf{LTL}^{\mathsf{Lim}}$ can be found in [7]. We prefer to give only an informal description here.

step 1 can be done nondeterministically in polynomial time, and steps 2–4 can be done (deterministically) in polynomial time. Furthermore, using LP, we also can check step 5 deterministically in polynomial time. For the lower-bound, we use [36] and note that if $\varphi = \top$ and $\beta = 0$, then the problem reduces to checking whether the underlying mp game has a Nash equilibrium. Based on the above observations, we have the following result.

▶ **Theorem 7.** Weak Implementation *with* $\mathsf{GR}(1)$ *specifications is* NP*-complete.*

Proof sketch. For the upper bound, we define an LP of size polynomial in $(\mathcal{G}, \kappa)$ having a solution if and only if there is an ultimately periodic path π such that $z_i \leq \mathsf{pay}_i(\pi)$ and satisfies the $\mathsf{GR}(1)$ specification. 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 subset of states in $(\mathcal{G}, \kappa)$ that satisfy the Boolean combinations ψ_l and θ_r, respectively. Property φ is satisfied on π if, and only if, either π visits every state in $V(\theta_r)$ infinitely often or some of the states in $V(\psi_l)$ only a finite number of times. For the game $(\mathcal{G}, \kappa)[z]$, let $W = (V, E, (\mathsf{w}_a)_{a \in \mathrm{N}})$ be the underlying multi-weighted graph, and for every edge $e \in E$ introduce a variable x_e. Informally, the value of x_e is the number of times that 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 $\mathsf{in}(v) = \{e \in E : \mathsf{trg}(e) = v\}$. Now, consider ψ_l for some $1 \leq l \leq m$, and define the following linear program $\mathsf{LP}(\psi_l)$:

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

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

Eq3: for each $a \in \mathrm{N}$, $\Sigma_{e \in E} \mathsf{w}_a(\mathsf{src}(e)) x_e \geq 0$ — total sum of any solution is non-negative;

Eq4: $\Sigma_{\mathsf{src}(e) \cap V(\psi_l) \neq \emptyset} x_e = 0$ — 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.

$\mathsf{LP}(\psi_l)$ has a solution if and only if there is 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.* Consider now the linear program $\mathsf{LP}(\theta_1, \ldots, \theta_n)$ defined as follows. Eq1–Eq3 as well as Eq5 are as in $\mathsf{LP}(\psi_l)$, and:

Eq4: for all $1 \leq r \leq n$, $\Sigma_{\mathsf{src}(e) \cap V(\theta_r) \neq \emptyset} x_e \geq 1$ — this condition says that, for every $V(\theta_r)$, at least one state in $V(\theta_r)$ is in the cycle associated with the solution of the linear program.

In this case, $\mathsf{LP}(\theta_1, \ldots, \theta_n)$ has 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 $(\mathcal{G}, \kappa)$ 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 [36] as discussed before. ◀

We now turn our attention to the strong implementation of the equilibrium design problem. As in this section, we first consider LTL specifications and then $\mathsf{GR}(1)$ specifications.

5 Equilibrium Design: Strong Implementation

Although the principal may find $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$ to be good news, it might not be good enough. It could be that even though there is a desirable Nash equilibrium, the others might be undesirable. This motivates us to consider the *strong implementation* variant of equilibrium design. Intuitively, in a strong implementation, we require that *every* Nash equilibrium outcome satisfies the specification φ, for a *non-empty* set of outcomes. Then, let $\mathrm{SI}(\mathcal{G}, \varphi, \beta)$ denote the set of subsidy schemes κ given budget β over $\mathcal{G}$ such that:

1. $(\mathcal{G}, \kappa)$ has at least one Nash equilibrium outcome,
2. every Nash equilibrium outcome of $(\mathcal{G}, \kappa)$ satisfies φ.

Formally we define it as follows:

$$\mathrm{SI}(\mathcal{G}, \varphi, \beta) = \{\kappa \in \mathcal{K}(\mathcal{G}, \beta) : \mathrm{NE}(\mathcal{G}, \kappa) \neq \varnothing \wedge \forall \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ s.t. } \pi(\vec{\sigma}) \models \varphi\}.$$

This gives us the following decision problem:

▶ **Definition 8** (Strong Implementation). *Given a game $\mathcal{G}$, formula φ, and budget β:*

Is it the case that $\mathrm{SI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$*?*

Strong Implementation can be solved with a 5-step procedure where the first four steps are as in Weak Implementation, and the last step (step 5) is as follows:

5 Check whether:

a. there is no ultimately periodic path π in $(\mathcal{G}, \kappa)[z]$ such that $z_i \leq \mathsf{pay}_i(\pi)$ for each $i \in \mathrm{N}$;
b. there is an ultimately periodic path π in $(\mathcal{G}, \kappa)[z]$ such that $\pi \models \neg\varphi$ and $z_i \leq \mathsf{pay}_i(\pi)$, for each $i \in \mathrm{N}$.

For step 5, observe that a positive answer to 5(a) or 5(b) is a counterexample to $\kappa \in \mathrm{SI}(\mathcal{G}, \varphi, \beta)$. Then, to carry out this procedure for the Strong Implementation problem with LTL specifications, consider the following $\mathsf{LTL}^{\mathsf{Lim}}$ formulae:

$$\varphi_{\exists} = \bigwedge_{i \in \mathrm{N}} (\mathsf{mp}(i) \geq z_i);$$

$$\varphi_{\forall} = \varphi_{\exists} \to \varphi.$$

Notice that the expression $\mathrm{NE}(\mathcal{G}, \kappa) \neq \varnothing$ can be expressed as "there exists a path π in $\mathcal{G}$ that satisfies formula $\varphi_{\exists}$". On the other hand, the expression $\forall \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa)$ such that $\pi(\vec{\sigma}) \models \varphi$ can be expressed as "for every path π in $\mathcal{G}$, if π satisfies formula $\varphi_{\exists}$, then π also satisfies formula φ". Thus, using these two formulae, we obtain the following result.

▶ **Corollary 9.** Strong Implementation *with* LTL *specifications is* PSPACE*-complete.*

Proof. Membership follows from the fact that step 5(a) can be solved by existential $\mathsf{LTL}^{\mathsf{Lim}}$ model checking, whereas step 5(b) by universal $\mathsf{LTL}^{\mathsf{Lim}}$ model checking—both clearly in PSPACE by Savitch's theorem. Hardness is similar to the construction in Theorem 6. ◀

Case with GR(1) specifications. Notice that the first part, *i.e.*, $\mathrm{NE}(\mathcal{G}, \kappa) \neq \varnothing$ can be solved in NP [36]. For the second part, observe that

$$\forall \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ such that } \pi(\vec{\sigma}) \models \varphi$$

is equivalent to

$$\neg \exists \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ such that } \pi(\vec{\sigma}) \models \neg\varphi.$$

Thus we have

$$\neg\varphi = \bigwedge_{l=1}^{m} \mathbf{GF}\psi_l \wedge \neg\left(\bigwedge_{r=1}^{n} \mathbf{GF}\theta_r\right).$$

To check this, we modify the LP in Theorem 7. Specifically, we modify Eq4 in $\mathsf{LP}(\theta_1, \ldots, \theta_n)$ to encode the θ-part of $\neg\varphi$. Thus, we have the following equation in $\mathsf{LP}'(\theta_1, \ldots, \theta_n)$:

Eq4: there exists r, $1 \le r \le n$, $\Sigma_{\mathsf{src}(e) \cap V(\theta_r) \neq \emptyset} x_e = 0$ — this condition ensures that at least one set $V(\theta_r)$ does not have any state in the cycle associated with the solution.

In this case, $\mathsf{LP}'(\theta_1, \ldots, \theta_n)$ has a solution if and only if there is a path π such that $z_i \le \mathsf{pay}_i(\pi)$ for every player i and, for at least one $V(\theta_r)$, its states are visited only *finitely many times*. Thus, we have a procedure that checks if there is a path π that satisfies $\neg\varphi$ such that $z_i \le \mathsf{pay}_i(\pi)$ for every player i, if and only if both linear programs have a solution. Using this new construction, we can now prove the following result.

▶ **Theorem 10.** Strong Implementation *with* $\mathsf{GR}(1)$ *specifications is* Σ_2^{P}*-complete.*

Proof sketch. For membership, observe that by rearranging the problem statement, we have the following question: Check whether the following expression is true

$$\exists \kappa \in \mathcal{K}(\mathcal{G}, \beta), \tag{1}$$

$$\exists \vec{\sigma} \in \sigma_1 \times \cdots \times \sigma_n, \text{ such that } \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa), \tag{2}$$

and

$$\forall \vec{\sigma}' \in \sigma_1 \times \cdots \times \sigma_n, \text{ if } \vec{\sigma}' \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ then } \pi(\vec{\sigma}') \models \varphi. \tag{3}$$

Statement (2) can be checked in NP (Theorem 4), whereas verifying statement (3) is in coNP; to see this, notice that we can rephrase (3) as follows: $\neg \exists z \in \{\mathsf{pun}_i(s) : s \in \mathrm{St}\}^{\mathrm{N}}$ such that both $\mathsf{LP}(\psi_l)$ and $\mathsf{LP}'(\theta_1, \ldots, \theta_n)$ have a solution in $(\mathcal{G}, \kappa)[z]$. Thus, membership in Σ_2^{P} follows. We prove hardness via a reduction from QSAT_2 (satisfiability of quantified Boolean formulae with 2 alternations), which is known to be Σ_2^{P}-complete [30]. ◀

6 Optimality and Uniqueness of Solutions

Having asked the questions studied in the previous sections, the principal – the *designer* in the equilibrium design problem – may want to explore further information. Because the power of the principal is limited by its budget, and because from the point of view of the system, it may be associated with a reward (*e.g.*, money, savings, etc.) or with the inverse of the amount of a finite resource (*e.g.*, time, energy, etc.) an obvious question is asking about *optimal* solutions. This leads us to *optimisation* variations of the problems we have studied. Informally, in this case, we ask what is the least budget that the principal needs to ensure that the implementation problems have positive solutions. The principal may also want to know whether a given subsidy scheme is *unique*, so that there is no point in looking for any other solutions to the problem. In this section, we investigate these kind of problems, and classify our study into two parts, one corresponding to the Weak Implementation problem and another one corresponding to the Strong Implementation problem.

6.1 Optimality and Uniqueness in the Weak Domain

We can now define formally some of the problems that we will study in the rest of this section. To start, the optimisation variant for Weak Implementation is defined as follows.

▶ **Definition 11** (Opt-WI). *Given a game $\mathcal{G}$ and a specification formula φ:*

What is the optimum budget β such that $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$?

Another natural problem, which is related to Opt-WI, is the "exact" variant – a membership question. In this case, in addition to $\mathcal{G}$ and φ, we are also given an integer b, and ask whether it is indeed the smallest amount of budget that the principal has to spend for some optimal weak implementation. This decision problem is formally defined as follows.

▶ **Definition 12** (Exact-WI). *Given a game $\mathcal{G}$, a specification formula φ, and an integer b:*

Is b equal to the optimum budget for $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$?

To study these problems, it is useful to introduce some concepts first. More specifically, let us introduce the concept of *implementation efficiency.* We say that a Weak Implementation (resp. Strong Implementation) is *efficient* if $\beta = \mathsf{cost}(\kappa)$ and there is no κ' such that $\mathsf{cost}(\kappa') < \mathsf{cost}(\kappa)$ and $\kappa' \in \mathrm{WI}(\mathcal{G}, \varphi, \beta)$ (resp. $\kappa' \in \mathrm{SI}(\mathcal{G}, \varphi, \beta)$). In addition to the concept of efficiency for an implementation problem, it is also useful to have the following result.

▶ **Proposition 13.** *Let z_i be the largest payoff that player i can get after deviating from a path π. The optimum budget is an integer between 0 and $\sum_{i \in \mathrm{N}} z_i \cdot (|\mathrm{St}| - 1)$.*

Using Proposition 13, we can show that both Opt-WI and Exact-WI can be solved in PSPACE for LTL specifications. Intuitively, the reason is that we can use the upper bound given by Proposition 13 to go through all possible solutions in exponential time, but using only nondeterministic polynomial space. Formally, we have the following results.

▶ **Theorem 14.** Opt-WI *with* LTL *specifications is* FPSPACE*-complete.*

▶ **Corollary 15.** Exact-WI *with* LTL *specifications is* PSPACE*-complete.*

The fact that both Opt-WI and Exact-WI with LTL specifications can be answered in, respectively, FPSPACE and PSPACE does not come as a big surprise: checking an instance can be done using polynomial space and there are only exponentially many instances to be checked. However, for Opt-WI and Exact-WI with GR(1) specifications, these two problems are more interesting.

▶ **Theorem 16.** Opt-WI *with* GR(1) *specifications is* $\mathsf{FP}^{\mathsf{NP}}$*-complete.*

Proof sketch. Membership follows from the fact that the search space, which is bounded as in Proposition 13, can be explored using binary search and Weak Implementation as an oracle. More precisely, we can find the smallest budget β such that $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$ by checking every possible value for β, which lies between 0 and 2^n, where n is the length of the encoding of the instance. Since, due to the binary search routine, we need logarithmically many calls to the NP oracle (*i.e.*, to Weak Implementation), in the end we have a searching procedure that would run in polynomial time. For the lower bound, we reduce from TSP Cost (the optimal travelling salesman problem), which is $\mathsf{FP}^{\mathsf{NP}}$-complete [30]. ◀

▶ **Corollary 17.** Exact-WI *with* GR(1) *specifications is* D^{P}*-complete.*

Proof. For membership, observe that an input is a "yes" instance of Exact-WI if and only if it is a "yes" instance of Weak Implementation *and* a "yes" instance of Weak Implementation Complement (the problem where one asks whether $\mathrm{WI}(\mathcal{G}, \varphi, \beta) = \varnothing$). Since the former problem is in NP and the latter problem is in coNP, membership in D^{P} follows. For the lower bound, we use the same reduction technique as in Theorem 16, and reduce from Exact TSP, a problem known to be D^{P}-hard [30, 31]. ◀

Following [29], we may naturally ask whether the optimal solution given by Opt-WI is unique. We call this problem UOpt-WI. For some fixed budget β, it may be the case that for two subsidy schemes $\kappa, \kappa' \in \mathrm{WI}(\mathcal{G}, \varphi, \beta)$ – we assume the implementation is efficient – we have $\kappa \neq \kappa'$ and $\mathsf{cost}(\kappa) = \mathsf{cost}(\kappa')$. With LTL specifications, it is not difficult to see that we can solve UOpt-WI in polynomial space. Therefore, we have the following result.

▶ **Corollary 18.** UOpt-WI *with* LTL *specifications is* PSPACE*-complete.*

For GR(1) specifications, we reason about UOpt-WI using the following procedure:

1. Find the exact budget using binary search and Weak Implementation as an oracle;
2. Use an NP oracle once to guess two distinct subsidy schemes with precisely this budget; if no such subsidy schemes exist, return "yes"; otherwise, return "no".

The above decision procedure clearly is in Δ_2^{P} (for the upper bound). Furthermore, since Theorem 16 implies Δ_2^{P}-hardness [23] (for the lower bound), we have the following corollary.

▶ **Corollary 19.** UOpt-WI *with* GR(1) *specifications is* Δ_2^{P}*-complete.*

6.2 Optimality and Uniqueness in the Strong Domain

In this subsection, we study the same problems as in the previous subsection but with respect to the Strong Implementation variant of the equilibrium design problem. We first formally define the problems of interest and then present the two first results.

▶ **Definition 20** (Opt-SI). *Given a game $\mathcal{G}$ and a specification formula φ:*

What is the optimum budget β such that $\mathrm{SI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$?

▶ **Definition 21** (Exact-SI). *Given a game $\mathcal{G}$, a specification formula φ, and an integer b:*

Is b equal to the optimum budget for $\mathrm{SI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$?

For the same reasons discussed in the weak versions of these two problems, we can prove the following two results with respect to games with LTL specifications.

▶ **Theorem 22.** Opt-SI *with* LTL *specifications is* FPSPACE*-complete.*

▶ **Corollary 23.** Exact-SI *with* LTL *specifications is* PSPACE*-complete.*

For GR(1) specifications, observe that using the same arguments for the upper-bound of Opt-WI with GR(1) specifications, we obtain the upper-bound for Opt-SI with GR(1) specifications. Then, it follows that Opt-SI is in $\mathsf{FP}^{\Sigma_2^{\mathsf{P}}}$. For hardness, we define an $\mathsf{FP}^{\Sigma_2^{\mathsf{P}}}$-complete problem, namely Weighted MinQSAT$_2$. Recall that in QSAT$_2$ we are given a Boolean 3DNF formula $\psi(\mathbf{x}, \mathbf{y})$ and sets $\mathbf{x} = \{x_1, \ldots, x_n\}, \mathbf{y} = \{y_1, \ldots, y_m\}$, with a set of terms $T = \{t_1, \ldots, t_k\}$. Define Weighted MinQSAT$_2$ as follows. Given $\psi(\mathbf{x}, \mathbf{y})$ and a weight function $\mathsf{c} : \mathbf{x} \to \mathbb{Z}^{\geq}$, Weighted MinQSAT$_2$ is the problem of finding an assignment $\vec{\mathbf{x}} \in \{0,1\}^n$ with the least total weight such that $\psi(\mathbf{x}, \mathbf{y})$ is true for every $\vec{\mathbf{y}} \in \{0,1\}^m$. Observe that Weighted MinQSAT$_2$ generalises MinQSAT$_2$, which is known to be $\mathsf{FP}^{\Sigma_2^{\mathsf{P}}[\log n]}$-hard [12], *i.e.*, MinQSAT$_2$ is an instance of Weighted MinQSAT$_2$, where all weights are 1.

▶ **Theorem 24.** Weighted MinQSAT$_2$ *is* $\mathsf{FP}^{\Sigma_2^\mathsf{P}}$*-complete.*

Proof. Membership follows from the upper-bound of MinQSAT$_2$ [12]: since we have an exponentially large input with respect to that of MinQSAT$_2$, by using binary search we will need polynomially many calls to the Σ_2^P oracle. Hardness is immediate [12]. ◀

Now that we have an $\mathsf{FP}^{\Sigma_2^\mathsf{P}}$-hard problem in our hands, we can proceed to determine the complexity class of Opt-SI with GR(1) specifications. For the upper bound we one can use arguments analogous to those in Theorem 16. For the lower bound, one can reduce from Weighted MinQSAT$_2$. Formally, we have:

▶ **Theorem 25.** Opt-SI *with* GR(1) *specifications is* $\mathsf{FP}^{\Sigma_2^\mathsf{P}}$*-complete.*

▶ **Corollary 26.** Exact-SI *with* GR(1) *specifications is* D_2^P*-complete.*

Proof. Membership follows from the fact that an input is a "yes" instance of Exact-SI (with GR(1) specifications) if and only if it is a "yes" instance of Strong Implementation *and* a "yes" instance of Strong Implementation Complement, the decision problem where we ask $\mathrm{SI}(\mathcal{G}, \varphi, \beta) = \varnothing$ instead. The lower bound follows from the hardness of Strong Implementation and Strong Implementation Complement problems, which immediately implies D_2^P-hardness [1, Lemma 3.2]. ◀

Furthermore, analogous to UOpt-WI, we also have the following corollaries.

▶ **Corollary 27.** UOpt-SI *with* LTL *specifications is* PSPACE*-complete.*

▶ **Corollary 28.** UOpt-SI *with* GR(1) *specifications is* Δ_3^P*-complete.*

7 Conclusions & Related and Future Work

Equilibrium design vs. mechanism design – connections with Economic theory.

Although equilibrium design is closely related to mechanism design, as typically studied in game theory [22], the two are not exactly the same. Two key features in mechanism design are the following. Firstly, in a mechanism design problem, the designer is not given a game structure, but instead is asked to provide one; in that sense, a mechanism design problem is closer to a rational synthesis problem [14, 17]. Secondly, in a mechanism design problem, the designer is only interested in the game's outcome, which is given by the payoffs of the players in the game; however, in equilibrium design, while the designer is interested in the payoffs of the players as these may need to be perturbed by its budget, the designer is also interested – and in fact primarily interested – in the satisfaction of a temporal logic goal specification, which the players in the game do not take into consideration when choosing their individual rational choices; in that sense, equilibrium design is closer to rational verification [18] than to mechanism design. Thus, equilibrium design is a new computational problem that sits somewhere in the middle between mechanism design and rational verification/synthesis. Technically, in equilibrium design we go beyond rational synthesis and verification through the additional design of subsidy schemes for incentivising behaviours in a concurrent and multi-agent system, but we do not require such subsidy schemes to be incentive compatible mechanisms, as in mechanism design theory, since the principal may want to reward only a group of players in the game so that its temporal logic goal is satisfied, while rewarding other players in the game in an unfair way – thus, leading to a game with a suboptimal social welfare measure. In this sense, equilibrium design falls short with respect to the more demanding social welfare requirements often found in mechanism design theory.

Equilibrium design vs. rational verification – connections with Computer science.

Typically, in rational synthesis and verification [14, 17, 18, 24] we want to check whether a property is satisfied on some/every Nash equilibrium computation run of a reactive, concurrent, and multi-agent system. These verification problems are primarily concerned with qualitative properties of a system, while assuming rationality of system components. However, little attention is paid to quantitative properties of the system. This drawback has been recently identified and some work has been done to cope with questions where both qualitative and quantitative concerns are considered [3, 6, 9, 10, 11, 19, 21, 37]. Equilibrium design is new and different approach where this is also the case. More specifically, as in a mechanism design problem, through the introduction of an external principal – the designer in the equilibrium design problem – we can account for overall qualitative properties of a system (the principal's goal given by an LTL or a GR(1) specification) as well as for quantitative concerns (optimality of solutions constrained by the budget to allocate additional rewards/resources). Our framework also mixes qualitative and quantitative features in a different way: while system components are only interested in maximising a quantitative payoff, the designer is primarily concerned about the satisfaction of a qualitative (logic) property of the system, and only secondarily about doing it in a quantitatively optimal way.

Equilibrium design vs. repair games and normative systems – connections with AI.

In recent years, there has been an interest in the analysis of rational outcomes of multi-agent systems modelled as multi-player games. This has been done both with modelling and with verification purposes. In those multi-agent settings, where AI agents can be represented as players in a multi-player game, a focus of interest is on the analysis of (Nash) equilibria in such games [8, 18]. However, it is often the case that the existence of Nash equilibria in a multi-player game with temporal logic goals may not be guaranteed [17, 18]. For this reason, there has been already some work on the introduction of desirable Nash equilibria in multi-player games [2, 32]. This problem has been studied as a repair problem [2] in which either the preferences of the players (given by winning conditions) or the actions available in the game are modified; the latter one also being achieved with the use of normative systems [32]. In equilibrium design, we do not directly modify the preferences of agents in the system, since we do not alter their goals or choices in the game, but we indirectly influence their rational behaviour by incentivising players to visit, or to avoid, certain states of the overall system. We studied how to do this in an (individually) optimal way with respect to the preferences of the principal in the equilibrium design problem. However, this may not always be possible, for instance, because the principal's temporal logic specification goal is just not achievable, or because of constraints given by its limited budget.

Future work: social welfare requirements and practical implementation.

As discussed before, a key difference with mechanism design is that social welfare requirements are not considered [26]. However, a benevolent principal might not see optimality as an individual concern, and instead consider the welfare of the players in the design of a subsidy scheme. In that case, concepts such as the *utilitarian social welfare* may be undesirable as the social welfare maximising the payoff received by players might allocate all the budget to only one player, and none to the others. A potentially better option is to improve fairness in the allocation of the budget by maximising the *egalitarian social welfare*. Finally, given that the complexity of equilibrium design is much better than that of rational synthesis/verification, we should be able to have efficient implementations, for instance, as an extension of EVE [20].

References

1 Gadi Aleksandrowicz, Hana Chockler, Joseph Y. Halpern, and Alexander Ivrii. The computational complexity of structure-based causality. *J. Artif. Int. Res.*, 58(1):431–451, January 2017.

2 S. Almagor, G. Avni, and O. Kupferman. Repairing Multi-Player Games. In *CONCUR*, volume 42 of *LIPIcs*, pages 325–339. Schloss Dagstuhl, 2015.

3 S. Almagor, O. Kupferman, and G. Perelli. Synthesis of controllable Nash equilibria in quantitative objective games. In *IJCAI*, pages 35–41, 2018.

4 B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded Strategy Logic: Reasoning about Uniqueness of Nash Equilibria. In *AAMAS*, pages 698–706. ACM, 2016.

5 R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa'ar. Synthesis of reactive(1) designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.

6 A. Bohy, V. Bruyère, E. Filiot, and J. Raskin. Synthesis from LTL Specifications with Mean-Payoff Objectives. In *TACAS*, pages 169–184, 2013.

7 U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal Specifications with Accumulative Values. *ACM Transactions on Computational Logic*, 15(4):27:1–27:25, 2014.

8 P. Bouyer, R. Brenguier, N. Markey, and M. Ummels. Pure Nash Equilibria in Concurrent Deterministic Games. *Logical Methods in Computer Science*, 11(2), 2015.

9 K. Chatterjee and L. Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60, 2012.

10 K. Chatterjee, L. Doyen, T. Henzinger, and J. Raskin. Generalized Mean-payoff and Energy Games. In *FSTTCS*, pages 505–516, 2010.

11 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-Payoff Parity Games. In *LICS*, pages 178–187. IEEE Computer Society, 2005.

12 H. Chockler and J. Halpern. Responsibility and Blame: A Structural-Model Approach. *Journal of Artificial Intelligence Research*, 22:93–115, 2004.

13 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Toruńczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and Helmut Veith, editors, *Computer Science Logic*, pages 260–274, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

14 D. Fisman, O. Kupferman, and Y. Lustig. Rational Synthesis. In *TACAS*, volume 6015 of *LNCS*, pages 190–204. Springer, 2010.

15 M. Garey, R. Graham, and D. Johnson. Some NP-complete Geometric Problems. In *STOC*, pages 10–22. ACM, 1976.

16 J. Gutierrez, P. Harrenstein, G. Perelli, and M. Wooldridge. Nash Equilibrium and Bisimulation Invariance. In *CONCUR*, volume 85 of *LIPIcs*, pages 17:1–17:16. Schloss Dagstuhl, 2017.

17 J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated Boolean Games. *Information and Computation*, 242:53–79, 2015.

18 J. Gutierrez, P. Harrenstein, and M. Wooldridge. From Model Checking to Equilibrium Checking: Reactive Modules for Rational Verification. *Artificial Intelligence*, 248:123–157, 2017.

19 J. Gutierrez, A. Murano, G. Perelli, S. Rubin, and M. Wooldridge. Nash Equilibria in Concurrent Games with Lexicographic Preferences. In *IJCAI*, pages 1067–1073, 2017.

20 J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. EVE: A Tool for Temporal Equilibrium Analysis. In *ATVA*, volume 11138 of *LNCS*, pages 551–557. Springer, 2018.

21 J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. On Computational Tractability for Rational Verification. In *IJCAI*, 2019. To appear.

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

23 M. Krentel. The Complexity of Optimization Problems. *Journal of Computer and System Sciences*, 36(3):490 – 509, 1988.

24 O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. *Annals of Mathematics and Artificial Intelligence*, 78(1):3–20, 2016.

25 M. Wooldridge and U. Endriss and S. Kraus and J. Lang. Incentive engineering for Boolean games. *Artificial Intelligence*, 195:418 – 439, 2013.

26 M. Maschler, E. Solan, and S. Zamir. *Game Theory*. Cambridge University Press, 2013.

27 M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.

28 C. Papadimitriou. The Euclidean Travelling Salesman Problem is NP-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.

29 C. Papadimitriou. On the Complexity of Unique Solutions. *Journal of the ACM*, 31(2):392–400, 1984.

30 C. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.

31 C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity). *Journal of Computer and System Sciences*, 28(2):244 – 259, 1984.

32 G. Perelli. Enforcing equilibria in multi-agent systems. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 188–196, 2019.

33 A. Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE, 1977.

34 A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *POPL*, pages 179–190. ACM Press, 1989.

35 S. Heubach and T. Mansour. *Combinatorics of Compositions and Words: Solutions Manual*. Chapman & Hall/CRC, 2009.

36 M. Ummels and D. Wojtczak. The Complexity of Nash Equilibria in Limit-Average Games. In *CONCUR*, pages 482–496, 2011.

37 Y. Velner, K. Chatterjee, L. Doyen, T. Henzinger, A. Rabinovich, and J. Raskin. The Complexity of Multi-Mean-Payoff and Multi-Energy Games. *Information and Computation*, 241:177–196, 2015.

38 U. Zwick and M. Paterson. The Complexity of Mean Payoff Games on Graphs. *Theoretical Computer Science*, 158(1):343 – 359, 1996.

A Proofs

Proof of Theorem 4

Proof. For (1) implies (2): Let z_i be the largest value player i can get by deviating from π. Let $k \in \mathbb{N}$ be such that $z_i = \mathsf{pun}_i(\mathsf{tr}(s_k, (\vec{\mathsf{a}}_{-i}, \mathsf{a}'_i)))$. Suppose further that $\mathsf{pay}_i(\pi) < z_i$. Thus, player i would deviate at s_k, which is a contradiction to π being a path induced by a Nash equilibrium.

For (2) imples (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. ◀

Proof of Proposition 5

Proof. For a fixed budget b, the number of subsidy schemes of budget exactly b corresponds to the number of *weak compositions* of b in m parts, which is given by $\binom{b+m-1}{b}$ [35]. Therefore, the number of subsidy schemes of budget at most β is the sum

$$|\mathcal{K}(\mathcal{G}, \beta)| = \sum_{b=0}^{\beta} \binom{b+m-1}{b}.$$

We now prove that

$$\sum_{b=0}^{\beta} \binom{b+m-1}{b} = \frac{\beta+1}{m}\binom{\beta+m}{\beta+1}.$$

By induction on β, as base case, for $\beta = 0$, we have that

$$\binom{\beta+m-1}{\beta} = 1 = \frac{\beta+1}{m}\binom{\beta+m}{\beta+1}.$$

For the inductive case, let us assume that the assertion hold for some β and let us prove for $\beta+1$. We have the following:

$$\sum_{b=0}^{\beta+1} \binom{b+m-1}{b} = \sum_{b=0}^{\beta} \binom{b+m-1}{b} + \binom{\beta+m-\cancel{1}+\cancel{1}}{\beta+1} = \frac{\beta+1}{m}\binom{\beta+m}{\beta+1} + \binom{\beta+m}{\beta+1}.$$

Therefore we have

$$\frac{\beta+1}{m}\binom{\beta+m}{\beta+1} + \binom{\beta+m}{\beta+1} = \binom{\beta+m}{\beta+1}\left(\frac{\beta+1}{m}+1\right) =$$

$$\binom{\beta+m}{\beta+1}\frac{\beta+1+m}{m} = \frac{\beta+1+m}{m} \cdot \frac{(\beta+m)!}{(\beta+1)!(\cancel{\beta}+m-\cancel{\beta}-1)!} =$$

$$\frac{(\beta+m+1)!}{(\beta+1)!m!} = \frac{(\beta+m+1)!}{(\beta+1)!m!} \cdot \frac{\beta+2}{\beta+2} \cdot \frac{m}{m} = \frac{\beta+2}{m} \cdot \frac{(\beta+m+1)!}{(\beta+2)!(m-1)!} =$$

$$\frac{\beta+2}{m} \cdot \frac{(\beta+m+1)!}{(\beta+2)!(\beta+m+1-\beta-2)!} = \frac{\beta+2}{m}\binom{\beta+m+1}{\beta+2}$$

that proves the assertion. ◀

Proof of Theorem 7

Proof. We will define a linear program of size polynomial in $(\mathcal{G}, \kappa)$ having a solution if and only if there exists an ultimately periodic path π such that $z_i \leq \mathsf{pay}_i(\pi)$ and satisfies the $\mathsf{GR}(1)$ specification.

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 subset of states in $(\mathcal{G}, \kappa)$ 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 $(\mathcal{G}, \kappa)[z]$, let $W = (V, E, (\mathsf{w}_a)_{a \in \mathrm{N}})$ be the underlying multi-weighted 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 non-negative;
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 $\mathsf{LP}(\theta_1, \ldots, \theta_n)$ 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 non-negative;
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, $\mathsf{LP}(\theta_1, \ldots, \theta_n)$ 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 $(\mathcal{G}, \kappa)$ 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 [36] and observe that if φ is true and $\beta = 0$, then the problem is equivalent to checking whether the mp game has a Nash equilibrium. ◀

Proof of Theorem 10

Proof. For membership, observe that by rearranging the problem statement, we have the following question:
Check whether the following expression is true

$$\exists \kappa \in \mathcal{K}(\mathcal{G}, \beta), \tag{1}$$

$$\exists \vec{\sigma} \in \sigma_1 \times \cdots \times \sigma_n, \text{ such that } \vec{\sigma} \in \mathrm{NE}(\mathcal{G}, \kappa), \tag{2}$$

and

$$\forall \vec{\sigma}' \in \sigma_1 \times \cdots \times \sigma_n, \text{ if } \vec{\sigma}' \in \mathrm{NE}(\mathcal{G}, \kappa) \text{ then } \pi(\vec{\sigma}') \models \varphi. \tag{3}$$

Statement (2) can be checked in NP (Theorem 4). Whereas, verifying statement (3) is in coNP; to see this, notice that we can rephrase (3) as follows: $\nexists z \in \{\mathtt{pun}_i(s) : s \in \mathrm{St}\}^{\mathrm{N}}$ such that both $\mathsf{LP}(\psi_l)$ and $\mathsf{LP}'(\theta_1, \dots, \theta_n)$ have a solution in $(\mathcal{G}, \kappa)[z]$. Thus Σ_2^{P} membership follows.

We prove hardness by a reduction from QSAT$_2$ (satisfiability of quantified Boolean formula with 2 alternations) [30]. Let $\psi(\mathbf{x}, \mathbf{y})$ be an $n + m$ variable Boolean 3DNF formula, where $\mathbf{x} = \{x_1, \dots, x_n\}$ and $\mathbf{y} = \{y_1, \dots, y_n\}$, with $t_1, \dots, t_k$ terms. Write $\mathbf{t}_j$ for the set of literals in j-th term and $\mathbf{t}_j^i$ for the i-th literal in $\mathbf{t}_j$. Moreover write x_i^j and y_i^j for variable $x_i \in \mathbf{x}$ and $y_i \in \mathbf{y}$ that appears in j-th term, respectively. For instance, if the fifth term is of the form of $(x_2 \wedge \neg x_3 \wedge y_4)$, then we have $\mathbf{t}_5 = \{x_2^5, x_3^5, y_4^5\}$ and $\mathbf{t}_5^1 = x_2^5$. Let $\mathbf{T} = \{\mathbf{t}_i \cap \mathbf{y} : 1 \leq i \leq k\}$, that is, the set of subset of $\mathbf{t}_i$ that contains only y-literals.

For a formula $\psi(\mathbf{x}, \mathbf{y})$ we construct an instance of Strong Implementation such that $\mathrm{SI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$ if and only if there is an $\vec{\mathbf{x}} \in \{0,1\}^n$ such that $\psi(\mathbf{x}, \mathbf{y})$ is true for every $\vec{\mathbf{y}} \in \{0,1\}^m$. Let $\mathcal{G}$ be such a game where

- $\mathrm{N} = \{1, 2\}$,
- $\mathrm{St} = \{\bigcup_{j \in [1,k]} (\mathbf{t}_j \times \{0,1\}^3)\} \cup \{\mathbf{T} \times \{0\}^3\} \cup \{\langle \mathtt{source}, \{0\}^3 \rangle, \langle \mathtt{sink}, \{0\}^3 \rangle\}$,
- $s_0 = \mathtt{source}$,
- for each state $s \in \mathrm{St}$
  - $\mathrm{Ac}_1(s) = \{\{\mathbf{T} \cup \{\mathtt{sink}\}\} \times \{0\}^3\}$, $\mathrm{Ac}_2(s) = \{\varepsilon\}$, if $s = \langle \mathtt{source}, \{0\}^3 \rangle$,
  - $\mathrm{Ac}_1(s) = \{\mathbf{t}_i^1 : s[0] \subseteq \mathbf{t}_i \wedge i \in [1,k]\}$, $\mathrm{Ac}_2(s) = \{0,1\}^3$, if $s \in \{\mathbf{T} \times \{0\}^3\}$,
  - $\mathrm{Ac}_1(s) = \{\varepsilon\}$, $\mathrm{Ac}_2(s) = \{\varepsilon\}$, if $s \in \bigcup_{j \in [1,k]} (\mathbf{t}_j \times \{0,1\}^3)$,
- for an action profile $\vec{\mathsf{a}} = (\mathsf{a}_1, \mathsf{a}_2)$
  - $\mathsf{tr}(s, \vec{\mathsf{a}}) = \mathsf{a}_1$, if $s = \langle \mathtt{source}, \{0\}^3 \rangle$,
  - $\mathsf{tr}(s, \vec{\mathsf{a}}) = \langle \mathsf{a}_1, \mathsf{a}_2 \rangle$, if $s \in \{\mathbf{T} \times \{0\}^3\}$,
  - $\mathsf{tr}(s, \vec{\mathsf{a}}) = \langle \mathbf{t}_j^{(i \bmod 3)+1}, s[1] \rangle$, if $s = \langle \mathbf{t}_j^i, s[1] \rangle \in \bigcup_{j \in [1,k]} (\mathbf{t}_j \times \{0,1\}^3)$;
  - $\mathsf{tr}(s, \vec{\mathsf{a}}) = s$, otherwise;
- for each state $s \in \mathrm{St}$, $\lambda(s) = s[0]$,
- for each state $s \in \mathrm{St}$
  - $\mathsf{w}_1(s) = \frac{2}{3}$, if $s[0] = \mathtt{sink}^3$,
  - $\mathsf{w}_1(s) = 0$, otherwise;
- the payoff of player $i \in \mathrm{N}$ for an ultimately periodic path π in $\mathcal{G}$ is
  - $\mathsf{pay}_1(\pi) = \mathsf{mp}(\mathsf{w}_1(\pi))$,

[3] This can be implemented by a macrostate with three substates—2 substates with weight of 1, and 1 with weight of 0—forming a simple cycle.

- $\mathsf{pay}_2(\pi) = -\mathsf{mp}(\mathsf{w}_1(\pi))$,

Furthermore, let $\beta = |\mathbf{x}|$ and the $\mathsf{GR}(1)$ property to be $\varphi := \mathbf{GF}\ \neg\mathtt{sink}$. Define a (partial) subsidy scheme $\kappa : \mathbf{x} \to \{0,1\}$. The weights are updated with respect to κ as follows: for each $s \in \mathrm{St}$ such that $s[0] \in \mathbf{t}_j \setminus \mathbf{y}$, that is, an x-literal that appears in term t_j

$$\mathsf{w}_1(s) = \begin{cases} 1, & \text{if } \kappa(s) = 1 \wedge s[0] \text{ is not negated in } t_j \\ 1, & \text{if } \kappa(s) = 0 \wedge s[0] \text{ is negated in } t_j \\ 0, & \text{if } \kappa(s) = 1 \wedge s[0] \text{ is negated in } t_j \\ 0, & \text{otherwise;} \end{cases}$$

for each $s \in \mathrm{St}$ such that $s[0] \in \mathbf{t}_j \setminus \mathbf{x}$, that is, a y-literal that appears in term t_j, $s[0] = \mathbf{t}_j^i$

$$\mathsf{w}_1(s) = \begin{cases} 1, & \text{if } s[1][i] = 1 \wedge s[0] \text{ is not negated in } t_j \\ 1, & \text{if } s[1][i] = 0 \wedge s[0] \text{ is negated in } t_j \\ 0, & \text{if } s[1][i] = 1 \wedge s[0] \text{ is negated in } t_j \\ 0, & \text{otherwise;} \end{cases}$$

the weights of other states remain unchanged.

The construction is now complete, and polynomial to the size of formula $\psi(\mathbf{x}, \mathbf{y})$. We claim that $\mathrm{SI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$ if and only if there is an $\vec{\mathbf{x}} \in \{0,1\}^n$ such that $\psi(\mathbf{x}, \mathbf{y})$ is true for every $\vec{\mathbf{y}} \in \{0,1\}^m$. From left to right, consider a subsidy scheme $\kappa \in \mathrm{SI}(\mathcal{G}, \varphi, \beta)$ which implies that there exists no Nash equilibrium run in $(\mathcal{G}, \kappa)$ that ends up in $\mathtt{sink}$. This means that for every action $\vec{\mathbf{a}}_2 \in \mathrm{Ac}_2(s)$, there exists $\vec{\mathbf{a}}_1 \in \mathrm{Ac}_1(s), s \in \{\mathbf{T} \times \{0\}^3\}$, such that $\mathsf{pay}_1(\pi) = 1$, where π is the resulting path of the joint action. Observe that this corresponds to the existence of (at least) a term t_i, which evaluates to true under assignment $\vec{\mathbf{x}}$, regardless the value of $\vec{\mathbf{y}}$. From right to left, consider an assigment $\vec{\mathbf{x}} \in \{0,1\}^n$ such that for all $\vec{\mathbf{y}} \in \{0,1\}^m$, the formula $\psi(\mathbf{x}, \mathbf{y})$ is true. This means that for every $\vec{\mathbf{y}}$, there exists (at least one) term t_i in $\psi(\mathbf{x}, \mathbf{y})$ that evaluates to true. By construction, specifically the weight updating rules, for every $\vec{\mathbf{a}}_2$ corresponding to assignment $\vec{\mathbf{y}}$, there exists $\mathbf{t}_j$ such that $\forall i \in [1,3], \mathsf{w}_1(\mathbf{t}_j^i) = 1$. This means that player 1 can always get payoff equals to 1, therefore, any run that ends in $\mathtt{sink}$ is not sustained by Nash equilibrium. ◀

Proof of Proposition 13

Proof. The lower-bound is straightforward. The upper-bound follows from the fact that the maximum value the principal has to pay to player i is when the path π is a simple cycle and formed from all states in St, apart from 1 deviation state. ◀

Proof of Theorem 14

Proof. Since the search space is bounded (Proposition 13), by using WEAK IMPLEMENTATION an an oracle we can iterate through every instance and return the smallest β such that $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$. Moreover, each instance is of polynomial size in the size of the input. Thus membership in PSPACE follows. Hardness is straightforward. ◀

Proof of Theorem 16

Proof. Membership follows from the fact that the search space, which is bounded as in Proposition 13, can be fully explored using binary search and WEAK IMPLEMENTATION as an oracle. More precisely, we can find the smallest budget β such that $\mathrm{WI}(\mathcal{G}, \varphi, \beta) \neq \varnothing$ by checking every possible value for β, which lies between 0 and 2^n, where n is the length of the encoding of the instance. Since we need logarithmically many calls to the NP oracle (to WEAK IMPLEMENTATION), in the end we have searching procedure that runs in polynomial time.

For hardness we reduce from TSP COST (optimal travelling salesman problem) that is known to be $\mathsf{FP}^{\mathsf{NP}}$-complete [30]. Given a TSP COST instance $\langle G, c\rangle$, $G = \langle V, E\rangle$ is a graph, $c : E \to \mathbb{Z}$ is a cost function. We assume that $\mathrm{WI}(\mathcal{G}, \varphi, \beta)$ is efficient. To encode TSP COST instance, we construct a game $\mathcal{G}$ and $\mathsf{GR}(1)$ formula φ, such that the optimum budget β corresponds to the value of optimum tour. Let $\mathcal{G}$ be such a game where

- $\mathrm{N} = \{1\}$,
- $\mathrm{St} = \{\langle v, e\rangle : v \in V \wedge e \in \mathsf{in}(v)\} \cup \{\langle \mathtt{sink}, \varepsilon\rangle\}$,
- s_0 can be chosen arbitrarily from $\mathrm{St} \setminus \{\langle \mathtt{sink}, \varepsilon\rangle\}$,
- for each state $\langle v, e\rangle \in \mathrm{St}$ and edge $e' \in E \cup \{\varepsilon\}$
  - $\mathsf{tr}(\langle v, e\rangle, e') = \langle \mathsf{trg}(e'), e'\rangle$, if $v \neq \mathtt{sink}$ and $e' \neq \varepsilon$,
  - $\mathsf{tr}(\langle v, e\rangle, e') = \langle \mathtt{sink}, \varepsilon\rangle$, otherwise;
- for each state $\langle v, e\rangle \in \mathrm{St}$
  - $\mathsf{w}_1(\langle v, e\rangle) = \max\{c(e') : e' \in E\} - c(e)$, if $v \neq \mathtt{sink}$,
  - $\mathsf{w}_1(\langle v, e\rangle) = \max\{c(e') : e' \in E\}$, otherwise;
- the payoff of player 1 for a path π in $\mathcal{G}$ is $\mathsf{pay}_1(\pi) = \mathsf{mp}(\mathsf{w}_1(\pi))$,
- for each state $\langle v, e\rangle \in \mathrm{St}$, the set of actions available to player 1 is $\mathsf{out}(v) \cup \{\varepsilon\}$,
- for each state $\langle v, e\rangle \in \mathrm{St}$, $\lambda(\langle v, e\rangle) = v$.

Furthermore, let $\varphi := \bigwedge_{v \in V} \mathbf{GF}\ v$. The construction is now complete, and is polynomial to the size of $\langle G, c\rangle$.

Now, consider the smallest $\mathsf{cost}(\kappa), \kappa \in \mathrm{WI}(\mathcal{G}, \varphi, \beta)$. We argue that $\mathsf{cost}(\kappa)$ is indeed the lowest value such that a tour in G is attainable. Suppose for contradiction, that there exists κ' such that $\mathsf{cost}(\kappa') < \mathsf{cost}(\kappa)$. Let π' be a path in $(\mathcal{G}, \kappa')$ and $z_1 = \mathsf{w}_1(\langle \mathtt{sink}, \varepsilon\rangle)$ the largest value player 1 can get by deviating from π'. We have $\mathsf{pay}_1(\pi') < z_1$, and since for every $\langle v, e\rangle \in \mathrm{St}$ there exists an edge to $\langle \mathtt{sink}, \varepsilon\rangle$, thus player 1 would deviate to $\langle \mathtt{sink}, \varepsilon\rangle$ and stay there forever. This deviation means that φ is not satisfied, which is a contradiction to $\kappa' \in \mathrm{WI}(\mathcal{G}, \varphi, \beta)$. The construction of φ also ensures that the path is a valid tour, i.e., the tour visits every city at least once. Notice that φ does not guarantee a Hamiltonian cycle. However, removing the condition of visiting each city *only once* does not remove the hardness, since *Euclidean* TSP is NP-hard [15, 28]. Therefore, in the planar case there is an optimal tour that visits each city only once, or otherwise, by the triangle inequality, skipping a repeated visit would not increase the cost. Finally, since $\mathrm{WI}(\mathcal{G}, \varphi, \beta)$ is efficient, we have β to be exactly the value of the optimum tour in the corresponding TSP COST instance. ◀

Proof of Theorem 22

Proof. The proof is analogous to that of Theorem 14. ◀

Proof of Theorem 25

Proof. Membership uses arguments analogous to those in Theorem 16. For hardness, we reduce WEIGHTED MINQSAT$_2$ to OPT-SI using the same techniques used in Theorem 10 with few modifications. Given a

Weighted MinQSAT$_2$ instance $\langle\psi(\mathbf{x},\mathbf{y}),\mathsf{c}\rangle$, we construct a game $\mathcal{G}$ and $\mathsf{GR}(1)$ formula φ, such that the optimum budget β corresponds to the value of optimal solution to $\langle\psi(\mathbf{x},\mathbf{y}),\mathsf{c}\rangle$. To this end, we may assume that $\mathrm{SI}(\mathcal{G},\varphi,\beta)$ is efficient and construct $\mathcal{G}$ with exactly the same rules as in Theorem 10 except for the following:

- clearly the value of β is unknown,
- the initial weight for each state $s \in \mathrm{St}$
  - $\mathsf{w}_1(s) = \frac{2}{3}$, if $s[0] = \mathtt{sink}$,
  - $$\mathsf{w}_1(s) = \begin{cases} -\mathsf{c}(s[0]) + 1, & \text{if } s[0] \in \mathbf{t}_j \setminus \mathbf{y} \wedge s[0] \text{ is not negated in } t_j \\ 1, & \text{if } s[0] \in \mathbf{t}_j \setminus \mathbf{y} \wedge s[0] \text{ is negated in } t_j; \end{cases}$$
  - $\mathsf{w}_1(s) = 0$, otherwise;
- given a subsidy scheme κ, we update the weight for each $s \in \mathrm{St}$ such that $s[0] \in \mathbf{t}_j \setminus \mathbf{y}$, that is, an x-literal that appears in term t_j

$$\mathsf{w}_1(s) = \begin{cases} \mathsf{w}_1(s) + \kappa(s), & \text{if } s[0] \text{ is not negated in } t_j \\ \mathsf{w}_1(s), & \text{otherwise;} \end{cases}$$

the construction is complete and polynomial to the size of $\langle\psi(\mathbf{x},\mathbf{y}),\mathsf{c}\rangle$.

Let o be the optimal solution to Weighted MinQSAT$_2$ given the input $\langle\psi(\mathbf{x},\mathbf{y}),\mathsf{c}\rangle$. We claim that β is exactly o. To see this, consider the smallest $\mathsf{cost}(\kappa)$, $\kappa \in \mathrm{SI}(\mathcal{G},\varphi,\beta)$. We argue that this is indeed the least total weight of an assignment $\vec{\mathbf{x}}$ such that $\psi(\mathbf{x},\mathbf{y})$ is true for every $\vec{\mathbf{y}}$. Assume towards a contradiction that $\mathsf{cost}(\kappa) < o$. By the construction of $\mathsf{w}_1(\cdot)$, there exists no π such that $\mathsf{pay}_1(\pi) > \frac{2}{3}$. Therefore, any run π' that ends up in $\mathtt{sink}$ is sustained by Nash equilibrium, which is a contradiction to $\kappa \in \mathrm{SI}(\mathcal{G},\varphi,\beta)$. Now, since $\mathrm{SI}(\mathcal{G},\varphi,\beta)$ is efficient, by definition, there exists no $\kappa' \in \mathrm{SI}(\mathcal{G},\varphi,\beta)$ such that $\mathsf{cost}(\kappa') < \mathsf{cost}(\kappa)$. Thus we have β equals to o as required. ◀

B Algorithms

Algorithm 1: WEAK IMPLEMENTATION.

```
1  Input: A game $\mathcal{G}$, a specification formula $\varphi$, and budget $\beta$.
2  for $\kappa \in \mathcal{K}(\mathcal{G}, \beta), s \in \mathrm{St}$, and $(\vec{\sigma}_{-1}, \ldots, \vec{\sigma}_{-n}) \in \bigtimes_{j \in \mathrm{N}} (\times_{i \in \mathrm{N} \setminus j} \sigma_i)$ do
3      Compute $(\mathcal{G}, \kappa)$
4      for $i \in \mathrm{N}$ do
5          Compute $z_i = \mathtt{pun}_i(s)$ using $\vec{\sigma}_{-i}$
6      Compute $(\mathcal{G}, \kappa)[z]$
7      if there is $\vec{\sigma} \in \mathrm{NE}((\mathcal{G}, \kappa)[z])$ such that $\pi(\vec{\sigma}) \models \varphi$ then
8          return Accept
9  return Reject
```

Algorithm 2: STRONG IMPLEMENTATION.

```
1  Input: A game $\mathcal{G}$, a specification formula $\varphi$, and budget $\beta$.
2  for $\kappa \in \mathcal{K}(\mathcal{G}, \beta)$ do
3      Compute $(\mathcal{G}, \kappa)$
4      for $i \in \mathrm{N}$ and $s \in \mathrm{St}$ do
5          Compute $\mathtt{pun}_i((\mathcal{G}, \kappa))$
6      $f_\exists \leftarrow \bot$; $f_\forall \leftarrow \top$
7      for $z \in \{\mathtt{pun}_i(s) : s \in \mathrm{St}\}^{\mathrm{N}}$ do
8          Compute $(\mathcal{G}, \kappa)[z]$
9          if there exists $\pi \in (\mathcal{G}, \kappa)[z]$ such that for each $i \in \mathrm{N}, \mathsf{pay}_i(\pi) \geq z_i$ then
10             $f_\exists \leftarrow \top$
11     for $z \in \{\mathtt{pun}_i(s) : s \in \mathrm{St}\}^{\mathrm{N}}$ do
12         if there exists $\pi \in (\mathcal{G}, \kappa)[z]$ such that for each $i \in \mathrm{N}, \mathsf{pay}_i(\pi) \geq z_i \wedge \pi \models \neg\varphi$
             then
13             $f_\forall \leftarrow \bot$
14     if $(f_\exists \wedge f_\forall)$ then
15         return Accept
16 return Reject
```