


# 1 Equilibrium Design for Concurrent Games

2 **Julian Gutierrez** 

3 Department of Computer Science, University of Oxford  
4 julian.gutierrez@cs.ox.ac.uk

5 **Muhammad Najib** 

6 Department of Computer Science, University of Oxford  
7 mnajib@cs.ox.ac.uk

8 **Giuseppe Perelli** 

9 Department of Computer Science, University of Göteborg  
10 giuseppe.perelli@gu.se

11 **Michael Wooldridge** 

12 Department of Computer Science, University of Oxford  
13 michael.wooldridge@cs.ox.ac.uk

## 14 — Abstract —

15 In game theory, *mechanism design* is concerned with the design of incentives so that a desired  
16 outcome of the game can be achieved. In this paper, we study the design of incentives so that  
17 a desirable equilibrium is obtained, for instance, an equilibrium satisfying a given temporal logic  
18 property—a problem that we call *equilibrium design*. We base our study on a framework where  
19 system specifications are represented as temporal logic formulae, games as quantitative concurrent  
20 game structures, and players’ goals as mean-payoff objectives. In particular, we consider system  
21 specifications given by LTL and GR(1) formulae, and show that implementing a mechanism to  
22 ensure that a given temporal logic property is satisfied on some/every Nash equilibrium of the game,  
23 whenever such a mechanism exists, can be done in PSPACE for LTL properties and in  $NP/\Sigma_2^P$  for  
24 GR(1) specifications. We also study the complexity of various related decision and optimisation  
25 problems, such as optimality and uniqueness of solutions, and show that the complexities of all such  
26 problems lie within the polynomial hierarchy. As an application, equilibrium design can be used as  
27 an alternative solution to the rational synthesis and verification problems for concurrent games with  
28 mean-payoff objectives whenever no solution exists, or as a technique to repair, whenever possible,  
29 concurrent games with undesirable rational outcomes (Nash equilibria) in an optimal way.

30 **2012 ACM Subject Classification** Theory of computation → Modal and temporal logics; Computing  
31 methodologies → Multi-agent systems; Theory of computation → Algorithmic game theory

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

33 **Digital Object Identifier** 10.4230/LIPIcs.CONCUR.2019.18

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

## 37 **1** Introduction

38 Over the past decade, there has been increasing interest in the use of game-theoretic  
39 equilibrium concepts such as Nash equilibrium in the analysis of concurrent and multi-agent  
40 systems (see, *e.g.*, [3, 4, 8, 14, 15, 17, 23]). This work views a concurrent system as a  
41 game, with system components (agents) corresponding to players in the game, which are  
42 assumed to be acting rationally in pursuit of their individual preferences. Preferences may  
43 be specified by associating with each player a temporal logic goal formula, which the player  
44 desires to see satisfied, or by assuming that players receive rewards in each state the system  
45 visits, and seek to maximise the average reward they receive (the *mean payoff*). A further



© Julian Gutierrez, Muhammad Najib, Giuseppe Perelli, Michael Wooldridge;  
licensed under Creative Commons License CC-BY

30th International Conference on Concurrency Theory (CONCUR 2019).

Editors: Wan Fokkink and Rob van Glabbeek; Article No. 18; pp. 18:1–18:16

Leibniz International Proceedings in Informatics



LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

46 possibility is to combine goals and rewards: players primarily seek the satisfaction of their  
 47 goal, and only secondarily seek to maximise their mean payoff. The key decision problems in  
 48 such settings relate to what temporal logic properties hold on computations of the system  
 49 that may be generated by players choosing strategies that form a game-theoretic (Nash)  
 50 equilibrium. These problems are typically computationally complex, since they subsume  
 51 temporal logic synthesis [32]. If players have LTL goals, for example, then checking whether  
 52 an LTL formula holds on some Nash equilibrium path in a concurrent game is 2EXPTIME-  
 53 complete [14, 16, 17], rather than only PSPACE-complete as it is the case for model checking,  
 54 certainly a computational barrier for the practical analysis and automated verification of  
 55 reactive, concurrent, and multi-agent systems modelled as multi-player games.

56 Within this game-theoretic reasoning framework, a key issue is that individually rational  
 57 choices can cause outcomes that are highly undesirable, and concurrent games also fall prey  
 58 to this problem. This has motivated the development of techniques for modifying games,  
 59 in order to avoid bad equilibria, or to facilitate good equilibria. *Mechanism design* is the  
 60 problem of designing a game such that, if players behave rationally, then a desired outcome  
 61 will be obtained [26]. Taxation and subsidy schemes are probably the most important class  
 62 of techniques used in mechanism design. They work by levying taxes on certain actions (or  
 63 providing subsidies), thereby incentivising players away from some outcomes towards others.  
 64 The present paper studies the design of subsidy schemes (incentives) for concurrent games,  
 65 so that a desired outcome (a Nash equilibrium in the game) can be obtained—a problem  
 66 that we call *Equilibrium design*. We model agents as synchronously executing concurrent  
 67 processes, with each agent receiving an integer payoff for every state the overall system visits;  
 68 the overall payoff an agent receives over an infinite computation path is then defined to be  
 69 the mean payoff over this path. While agents (naturally) seek to maximise their individual  
 70 mean payoff, the designer of the subsidy scheme wishes to see some temporal logic formula  
 71 satisfied, either on some or on every Nash equilibrium of the game.

72 With this model, we assume that the designer – an external principal – has a finite budget  
 73 that is available for making subsidies, and this budget can be allocated across agent/state  
 74 pairs. By allocating this budget appropriately, the principal can incentivise players away from  
 75 some states and towards others. Since the principal has some temporal logic goal formula, it  
 76 desires to allocate subsidies so that players are rationally incentivised to choose strategies so  
 77 that the principal’s temporal logic goal formula is satisfied in the path that would result from  
 78 executing the strategies. For this general problem, following [24], we identify two variants of  
 79 the principal’s mechanism design problem, which we refer to as WEAK IMPLEMENTATION  
 80 and STRONG IMPLEMENTATION. In the WEAK variant, we ask whether the principal can  
 81 allocate the budget so that the goal is achieved on *some* computation path that would be  
 82 generated by Nash equilibrium strategies in the resulting system; in the STRONG variation,  
 83 we ask whether the principal can allocate the budget so that the resulting system has at least  
 84 one Nash equilibrium, and moreover the temporal logic goal is satisfied on *all* paths that  
 85 could be generated by Nash equilibrium strategies. For these two problems, we consider goals  
 86 specified by LTL formulae or GR(1) formulae [5], give algorithms for each case, and classify the  
 87 complexity of the problem. While LTL is a natural language for the specification of properties  
 88 of concurrent and multi-agent systems, GR(1) is an LTL fragment that can be used to easily  
 89 express several prefix-independent properties of computation paths of reactive systems, such  
 90 as  $\omega$ -regular properties often used in automated formal verification. We then go on to  
 91 study variations of these two problems, for example considering *optimality* and *uniqueness*  
 92 of solutions, and show that the complexities of all such problems lie within the polynomial  
 93 hierarchy, thus making them potentially amenable to efficient practical implementations.

94 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)	$\Sigma_2^P$ -complete (Thm. 10)
OPT-WI	FSPACE-complete (Thm. 14)	$\text{FP}^{\text{NP}}$ -complete (Thm. 16)
OPT-SI	FSPACE-complete (Thm. 22)	$\text{FP}^{\Sigma_2^P}$ -complete (Thm. 25)
EXACT-WI	PSPACE-complete (Cor. 15)	$\text{D}^P$ -complete (Cor. 17)
EXACT-SI	PSPACE-complete (Cor. 23)	$\text{D}_2^P$ -complete (Cor. 26)
UOPT-WI	PSPACE-complete (Cor. 18)	$\Delta_2^P$ -complete (Cor. 19)
UOPT-SI	PSPACE-complete (Cor. 27)	$\Delta_3^P$ -complete (Cor. 28)

■ **Table 1** Summary of main complexity results.

## 95 2 Preliminaries

**Linear Temporal Logic.** LTL [31] 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:

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

96 where  $p \in \text{AP}$ . As commonly found in the LTL literature, we use of the following abbreviations:  
 97  $\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2)$ ,  $\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2$ ,  $\mathbf{F}\varphi \equiv \top \mathbf{U} \varphi$ , and  $\mathbf{G}\varphi \equiv \neg\mathbf{F}\neg\varphi$ .

We interpret formulae of LTL with respect to pairs  $(\alpha, t)$ , where  $\alpha \in (2^{\text{AP}})^\omega$  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  $\alpha$ . Formally, the semantics of LTL formulae is given by the following rules:

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

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

**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]:

$$(\mathbf{GF}\psi_1 \wedge \dots \wedge \mathbf{GF}\psi_m) \rightarrow (\mathbf{GF}\varphi_1 \wedge \dots \wedge \mathbf{GF}\varphi_n),$$

99 where each subformula  $\psi_i$  and  $\varphi_i$  is a Boolean combination of atomic propositions.

## 18:4 Equilibrium Design for Concurrent Games

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

$$\text{mp}(r) = \liminf_{n \rightarrow \infty} \text{avg}_n(r)$$

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

101 **Arenas.** An *arena* is a tuple  $A = \langle N, \text{Ac}, \text{St}, s_0, \text{tr}, \lambda \rangle$  where  $N$ ,  $\text{Ac}$ , and  $\text{St}$  are finite non-  
102 empty sets of *players* (write  $N = |N|$ ), *actions*, and *states*, respectively; if needed, we write  
103  $\text{Ac}_i(s)$ , to denote the set of actions available to player  $i$  at  $s$ ;  $s_0 \in \text{St}$  is the *initial state*;  
104  $\text{tr} : \text{St} \times \vec{\text{Ac}} \rightarrow \text{St}$  is a *transition function* mapping each pair consisting of a state  $s \in \text{St}$  and  
105 an *action profile*  $\vec{a} \in \vec{\text{Ac}} = \text{Ac}^N$ , one for each player, to a successor state; and  $\lambda : \text{St} \rightarrow 2^{\text{AP}}$   
106 is a labelling function, mapping every state to a subset of *atomic propositions*.

107 We sometimes call an action profile  $\vec{a} = (\mathbf{a}_1, \dots, \mathbf{a}_n) \in \vec{\text{Ac}}$  a *decision*, and denote  $\mathbf{a}_i$  the  
108 action taken by player  $i$ . We also consider *partial* decisions. For a set of players  $C \subseteq N$  and  
109 action profile  $\vec{a}$ , we let  $\vec{a}_C$  and  $\vec{a}_{-C}$  be two tuples of actions, respectively, one for all players  
110 in  $C$  and one for all players in  $N \setminus C$ . We also write  $\vec{a}_i$  for  $\vec{a}_{\{i\}}$  and  $\vec{a}_{-i}$  for  $\vec{a}_{N \setminus \{i\}}$ . For two  
111 decisions  $\vec{a}$  and  $\vec{a}'$ , we write  $(\vec{a}_C, \vec{a}'_{-C})$  to denote the decision where the actions for players in  
112  $C$  are taken from  $\vec{a}$  and the actions for players in  $N \setminus C$  are taken from  $\vec{a}'$ .

113 A *path*  $\pi = (s_0, \vec{a}^0), (s_1, \vec{a}^1) \dots$  is an infinite sequence in  $(\text{St} \times \vec{\text{Ac}})^\omega$  such that  $\text{tr}(s_k, \vec{a}^k) =$   
114  $s_{k+1}$  for all  $k$ . Paths are generated in the arena by each player  $i$  selecting a *strategy*  $\sigma_i$  that  
115 will define how to make choices over time. We model strategies as finite state machines with  
116 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  
117 machine with output (a transducer), where  $Q_i$  is a finite and non-empty set of *internal states*,  
118  $q_i^0$  is the *initial state*,  $\delta_i : Q_i \times \vec{\text{Ac}} \rightarrow Q_i$  is a deterministic *internal transition function*, and  
119  $\tau_i : Q_i \rightarrow \text{Ac}_i$  an *action function*. Let  $\text{Str}_i$  be the set of strategies for player  $i$ . Note that this  
120 definition implies that strategies have *perfect information*<sup>1</sup> and finite memory (although we  
121 impose no bounds on memory size).

122 A *strategy profile*  $\vec{\sigma} = (\sigma_1, \dots, \sigma_n)$  is a vector of strategies, one for each player. As with  
123 actions,  $\vec{\sigma}_i$  denotes the strategy assigned to player  $i$  in profile  $\vec{\sigma}$ . Moreover, by  $(\vec{\sigma}_B, \vec{\sigma}'_C)$   
124 we denote the combination of profiles where players in disjoint  $B$  and  $C$  are assigned their  
125 corresponding strategies in  $\vec{\sigma}$  and  $\vec{\sigma}'$ , respectively. Once a state  $s$  and profile  $\vec{\sigma}$  are fixed, the  
126 game has an *outcome*, a path in  $A$ , denoted by  $\pi(\vec{\sigma}, s)$ . Because strategies are deterministic,  
127  $\pi(\vec{\sigma}, s)$  is the unique path induced by  $\vec{\sigma}$ , that is, the sequence  $s_0, s_1, s_2, \dots$  such that

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

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

131 Arenas define the dynamic structure of games, but lack a central aspect of a game:  
132 preferences, which give games their strategic structure. A *multi-player game* is obtained  
133 from an arena  $A$  by associating each player with a goal. We consider multi-player games  
134 with **mp** goals. A multi-player **mp** game is a tuple  $\mathcal{G} = \langle A, (\mathbf{w}_i)_{i \in N} \rangle$ , where  $A$  is an arena and  
135  $\mathbf{w}_i : \text{St} \rightarrow \mathbb{Z}$  is a function mapping, for every player  $i$ , every state of the arena into an integer  
136 number. In any game with arena  $A$ , a path  $\pi$  in  $A$  induces a sequence  $\lambda(\pi) = \lambda(s_0)\lambda(s_1) \dots$   
137 of sets of atomic propositions; if, in addition,  $A$  is the arena of an **mp** game, then, for each  
138 player  $i$ , the sequence  $\mathbf{w}_i(\pi) = \mathbf{w}_i(s_0)\mathbf{w}_i(s_1) \dots$  of weights is also induced. Unless stated  
139 otherwise, for a game  $\mathcal{G}$  and a path  $\pi$  in it, the payoff of player  $i$  is  $\text{pay}_i(\pi) = \text{mp}(\mathbf{w}_i(\pi))$ .

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

<sup>1</sup> Mean-payoff games with imperfect information are generally undecidable [13].

Nash equilibrium [26]. 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  $\sigma'_i$  for player  $i$ , we have

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

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

### 141 **3 From Mechanism Design to Equilibrium Design**

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

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

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

165 **► Definition 1.** Given a game  $\mathcal{G}$  and subsidy scheme  $\kappa$ , we let  $\text{cost}(\kappa) = \sum_{i \in \mathbb{N}} \sum_{s \in \text{St}} \kappa_i(s)$ .

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

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

173 We now come to the main question(s) that we consider in the remainder of the paper.  
174 We ask whether the principal can find a subsidy scheme that will incentivise players to  
175 collectively choose a rational outcome (a Nash equilibrium) that satisfies its temporal logic  
176 goal  $\varphi$ . We call this problem *equilibrium design*. Following [24], we define two variants of this  
177 problem, a *weak* and a *strong* implementation of the equilibrium design problem. The formal  
178 definition of the problems and the analysis of their respective computational complexity are  
179 presented in the next sections.

## 180 **4** Equilibrium Design: Weak Implementation

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

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

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

188 **► Definition 3** (WEAK IMPLEMENTATION). *Given a game  $\mathcal{G}$ , formula  $\varphi$ , and budget  $\beta$ :*

189 *Is it the case that  $\text{WI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$ ?*

190 In order to solve WEAK IMPLEMENTATION, we first characterise the Nash equilibria of a  
 191 multi-player concurrent game in terms of punishment strategies. To do this in our setting,  
 192 we recall the notion of secure values for mean-payoff games [33].

193 For a player  $i$  and a state  $s \in \text{St}$ , by  $\text{pun}_i(s)$  we denote the punishment value of  $i$  over  
 194  $s$ , that is, the maximum payoff that  $i$  can achieve from  $s$ , when all other players behave  
 195 adversarially. Such a value can be computed by considering the corresponding two-player  
 196 zero-sum mean-payoff game [35]. Thus, it is in  $\text{NP} \cap \text{coNP}$ , and note that both player  $i$  and  
 197 coalition  $N \setminus \{i\}$  can achieve the optimal value of the game using *memoryless* strategies. Then,  
 198 for a player  $i$  and a value  $z \in \mathbb{R}$ , a pair  $(s, \vec{\mathbf{a}})$  is  $z$ -secure for player  $i$  if  $\text{pun}_i(\text{tr}(s, (\vec{\mathbf{a}}_{-i}, \mathbf{a}'_i))) \leq z$   
 199 for every  $\mathbf{a}'_i \in \text{Ac}$ . Write  $\text{pun}_i(\mathcal{G})$  for the set of punishment values for player  $i$  in  $\mathcal{G}$ .

200 **► Theorem 4.** *For every mp game  $\mathcal{G}$  and ultimately periodic path  $\pi = (s_0, \vec{\mathbf{a}}_0), (s_1, \vec{\mathbf{a}}_1), \dots$ ,*  
 201 *the following are equivalent:*

- 202 1. *There is  $\vec{\sigma} \in \text{NE}(\mathcal{G})$  such that  $\pi = \pi(\vec{\sigma}, s_0)$ ;*
- 203 2. *There exists  $z \in \mathbb{R}^N$ , where  $z_i \in \text{pun}_i(\mathcal{G})$  such that, for every  $i \in N$* 
  - 204 a. *for all  $k \in \mathbb{N}$ , the pair  $(s_k, \vec{\mathbf{a}}^k)$  is  $z_i$ -secure for  $i$ , and*
  - 205 b.  *$z_i \leq \text{pay}_i(\pi)$ .*

206 The characterisation of Nash Equilibria provided in Theorem 4 will allow us to turn the  
 207 WEAK IMPLEMENTATION problem into a *path finding* problem over  $(\mathcal{G}, \kappa)$ . On the other  
 208 hand, with respect to the budget  $\beta$  that the principal has at its disposal, the definition of  
 209 subsidy scheme function  $\kappa$  implies that the size of  $\mathcal{K}(\mathcal{G}, \beta)$  is bounded, and particularly, it is  
 210 bounded by  $\beta$  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  $|\text{St}|$  states and budget  $\beta$ , it holds  
 that*

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

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

212 From Proposition 5 we derive that the number of possible subsidy schemes is *polynomial*  
 213 in the budget  $\beta$  and singly *exponential* in both the number of agents and states in the game.  
 214 At this point, solving WEAK IMPLEMENTATION can be done with the following procedure:

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

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  $(\vec{\sigma}_{-1}, \dots, \vec{\sigma}_{-n})$ , we can compute  $z$  solving  $|\mathbb{N}|$  one-player mean-payoff games, one for each player  $i$  [35, 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

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

225 written in  $\text{LTL}^{\text{Lim}}$  [7], an extension of LTL where statements about mean-payoff values over

226 a given weighted arena can be made.<sup>2</sup> The semantics of the temporal operators of  $\text{LTL}^{\text{Lim}}$

227 is just like the one for LTL over infinite computation paths  $\pi = s_0, s_1, s_3, \dots$ . On the other

228 hand, the meaning of  $\text{mp}(i) \geq z_i$  is simply that such an atomic formula is true if, and only if,

229 the mean-payoff value of  $\pi$  with respect to player  $i$  is greater or equal to  $z_i$ , a constant real

230 value; that is,  $\text{mp}(i) \geq z_i$  is true in  $\pi$  if and only if  $\text{pay}_i(\pi) = \text{mp}(\mathbf{w}_i(\pi))$  is greater or equal

231 than constant value  $z_i$ . Formula  $\varphi_{\text{WI}}$  corresponds exactly to 2(b) in Theorem 4. Furthermore,

232 since every path in  $(\mathcal{G}, \kappa)[z]$  satisfies condition 2(a) of Theorem 4, every computation path of

233  $(\mathcal{G}, \kappa)[z]$  that satisfies  $\varphi_{\text{WI}}$  is a witness to the WEAK IMPLEMENTATION problem.

234 ► **Theorem 6.** WEAK IMPLEMENTATION with LTL specifications is PSPACE-complete.

235 **Proof.** Membership follows from the procedure above and the fact that model checking for

236  $\text{LTL}^{\text{Lim}}$  is PSPACE-complete [7]. Hardness follows from the fact that LTL model checking is a

237 special case of WEAK IMPLEMENTATION. For instance, consider the case in which all weights

238 for all players are set to the same value, say 0, and the principal has budget  $\beta = 0$ . ◀

239 **Case with GR(1) specifications.** One of the main bottlenecks of our procedure to solve

240 WEAK IMPLEMENTATION lies in step 5, where we solve an  $\text{LTL}^{\text{Lim}}$  model checking problem.

241 To reduce the complexity of our decision procedure, we consider WEAK IMPLEMENTATION

242 with the specification  $\varphi$  expressed in the GR(1) sublanguage of LTL. With this specification

243 language, the path finding problem can be solved without model-checking the  $\text{LTL}^{\text{Lim}}$  formula

244 given before. In order to do this, we can define a linear program (LP) such that the LP has

245 a solution if and only if  $\text{WI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$ . From our previous procedure, observe that

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

## 18:8 Equilibrium Design for Concurrent Games

246 step 1 can be done nondeterministically in polynomial time, and steps 2–4 can be done  
 247 (deterministically) in polynomial time. Furthermore, using LP, we also can check step 5  
 248 deterministically in polynomial time. For the lower-bound, we use [33] and note that if  
 249  $\varphi = \top$  and  $\beta = 0$ , then the problem reduces to checking whether the underlying **mp** game  
 250 has a Nash equilibrium. Based on the above observations, we have the following result.

251 ► **Theorem 7.** WEAK IMPLEMENTATION *with 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  $\pi$  such that  $z_i \leq \text{pay}_i(\pi)$  and satisfies the GR(1) specification. Recall that  $\varphi$  has the following form

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

252 and let  $V(\psi_l)$  and  $V(\theta_r)$  be the subset of states in  $(\mathcal{G}, \kappa)$  that satisfy the Boolean combinations  
 253  $\psi_l$  and  $\theta_r$ , respectively. Property  $\varphi$  is satisfied on  $\pi$  if, and only if, either  $\pi$  visits every state  
 254 in  $V(\theta_r)$  infinitely often or some of the states in  $V(\psi_l)$  only a finite number of times. For  
 255 the game  $(\mathcal{G}, \kappa)[z]$ , let  $W = (V, E, (\mathbf{w}_a)_{a \in N})$  be the underlying multi-weighted graph, and  
 256 for every edge  $e \in E$  introduce a variable  $x_e$ . Informally, the value of  $x_e$  is the number of  
 257 times that  $e$  is used on a cycle. Formally, let  $\text{src}(e) = \{v \in V : \exists w e = (v, w) \in E\}$ ;  $\text{trg}(e) =$   
 258  $\{v \in V : \exists w e = (w, v) \in E\}$ ;  $\text{out}(v) = \{e \in E : \text{src}(e) = v\}$ ; and  $\text{in}(v) = \{e \in E : \text{trg}(e) = v\}$ .  
 259 Now, consider  $\psi_l$  for some  $1 \leq l \leq m$ , and define the following linear program  $\text{LP}(\psi_l)$ :

260Eq1:  $x_e \geq 0$  for each edge  $e$  — a basic consistency criterion;

261Eq2:  $\sum_{e \in E} x_e \geq 1$  — at least one edge is chosen;

262Eq3: for each  $a \in N$ ,  $\sum_{e \in E} \mathbf{w}_a(\text{src}(e)) x_e \geq 0$  — total sum of any solution is non-negative;

263Eq4:  $\sum_{\text{src}(e) \cap V(\psi_l) \neq \emptyset} x_e = 0$  — no state in  $V(\psi_l)$  is in the cycle associated with the solution;

264Eq5: for each  $v \in V$ ,  $\sum_{e \in \text{out}(v)} x_e = \sum_{e \in \text{in}(v)} x_e$  — this condition says that the number of times  
 265 one enters a vertex is equal to the number of times one leaves that vertex.

266  $\text{LP}(\psi_l)$  has a solution if and only if there is a path  $\pi$  in  $\mathcal{G}$  such that  $z_i \leq \text{pay}_i(\pi)$  for  
 267 every player  $i$  and visits  $V(\psi_l)$  only *finitely many times*. Consider now the linear program  
 268  $\text{LP}(\theta_1, \dots, \theta_n)$  defined as follows. Eq1–Eq3 as well as Eq5 are as in  $\text{LP}(\psi_l)$ , and:

269Eq4: for all  $1 \leq r \leq n$ ,  $\sum_{\text{src}(e) \cap V(\theta_r) \neq \emptyset} x_e \geq 1$  — this condition says that, for every  $V(\theta_r)$ , at  
 270 least one state in  $V(\theta_r)$  is in the cycle associated with the solution of the linear program.

271 In this case,  $\text{LP}(\theta_1, \dots, \theta_n)$  has a solution if and only if there exists a path  $\pi$  such that  
 272  $z_i \leq \text{pay}_i(\pi)$  for every player  $i$  and visits every  $V(\theta_r)$  *infinitely many times*. Since the  
 273 constructions above are polynomial in the size of both  $(\mathcal{G}, \kappa)$  and  $\varphi$ , we can conclude it is  
 274 possible to check in NP the statement that there is a path  $\pi$  satisfying  $\varphi$  such that  $z_i \leq \text{pay}_i(\pi)$   
 275 for every player  $i$  in the game if and only if one of the two linear programs defined above has  
 276 a solution. For the lower-bound, we use [33] as discussed before. ◀

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



## 5 Equilibrium Design: Strong Implementation

Although the principal may find  $WI(\mathcal{G}, \varphi, \beta) \neq \emptyset$  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  $\varphi$ , for a *non-empty* set of outcomes. Then, let  $SI(\mathcal{G}, \varphi, \beta)$  denote the set of subsidy schemes  $\kappa$  given budget  $\beta$  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  $\varphi$ .

Formally we define it as follows:

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

This gives us the following decision problem:

► **Definition 8** (STRONG IMPLEMENTATION). *Given a game  $\mathcal{G}$ , formula  $\varphi$ , and budget  $\beta$ :*

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

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  $\pi$  in  $(\mathcal{G}, \kappa)[z]$  such that  $z_i \leq \text{pay}_i(\pi)$  for each  $i \in \mathbb{N}$ ;
- b. there is an ultimately periodic path  $\pi$  in  $(\mathcal{G}, \kappa)[z]$  such that  $\pi \models \neg\varphi$  and  $z_i \leq \text{pay}_i(\pi)$ , for each  $i \in \mathbb{N}$ .

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

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

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

Notice that the expression  $NE(\mathcal{G}, \kappa) \neq \emptyset$  can be expressed as “there exists a path  $\pi$  in  $\mathcal{G}$  that satisfies formula  $\varphi_{\exists}$ ”. On the other hand, the expression  $\forall \vec{\sigma} \in NE(\mathcal{G}, \kappa)$  such that  $\pi(\vec{\sigma}) \models \varphi$  can be expressed as “for every path  $\pi$  in  $\mathcal{G}$ , if  $\pi$  satisfies formula  $\varphi_{\exists}$ , then  $\pi$  also satisfies formula  $\varphi$ ”. 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  $LTL^{\text{Lim}}$  model checking, whereas step 5(b) by universal  $LTL^{\text{Lim}}$  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.*,  $NE(\mathcal{G}, \kappa) \neq \emptyset$  can be solved in NP [33]. For the second part, observe that

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

## 18:10 Equilibrium Design for Concurrent Games

is equivalent to

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

Thus we have

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

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

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

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

320 ► **Theorem 10.** STRONG IMPLEMENTATION with GR(1) specifications is  $\Sigma_2^P$ -complete.

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

$$323 \quad \exists \kappa \in \mathcal{K}(\mathcal{G}, \beta), \tag{1}$$

$$324 \quad \exists \vec{\sigma} \in \sigma_1 \times \dots \times \sigma_n, \text{ such that } \vec{\sigma} \in \text{NE}(\mathcal{G}, \kappa), \tag{2}$$

325 and

$$326 \quad \forall \vec{\sigma}' \in \sigma_1 \times \dots \times \sigma_n, \text{ if } \vec{\sigma}' \in \text{NE}(\mathcal{G}, \kappa) \text{ then } \pi(\vec{\sigma}') \models \varphi. \tag{3}$$

328 Statement (2) can be checked in NP (Theorem 4), whereas verifying statement (3) is in  
 329 coNP; to see this, notice that we can rephrase (3) as follows:  $\neg \exists z \in \{\text{pun}_i(s) : s \in \text{St}\}^N$  such  
 330 that both  $\text{LP}(\psi_l)$  and  $\text{LP}'(\theta_1, \dots, \theta_n)$  have a solution in  $(\mathcal{G}, \kappa)[z]$ . Thus, membership in  $\Sigma_2^P$   
 331 follows. We prove hardness via a reduction from  $\text{QSAT}_2$  (satisfiability of quantified Boolean  
 332 formulae with 2 alternations), which is known to be  $\Sigma_2^P$ -complete [28]. ◀

## 333 6 Optimality and Uniqueness of Solutions

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

*What is the optimum budget  $\beta$  such that  $\text{WI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$ ?*

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  $\varphi$ , 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  $\varphi$ , and an integer  $b$ :*

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

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  $\beta = \text{cost}(\kappa)$  and there is no  $\kappa'$  such that  $\text{cost}(\kappa') < \text{cost}(\kappa)$  and  $\kappa' \in \text{WI}(\mathcal{G}, \varphi, \beta)$  (resp.  $\kappa' \in \text{SI}(\mathcal{G}, \varphi, \beta)$ ). 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  $\pi$ . The optimum budget is an integer between 0 and  $\sum_{i \in N} z_i \cdot (|\text{St}| - 1)$ .*

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  $\text{FP}^{\text{NP}}$ -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  $\beta$  such that  $\text{WI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$  by checking every possible value for  $\beta$ , 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  $\text{FP}^{\text{NP}}$ -complete [28]. ◀

► **Corollary 17.** *EXACT-WI with GR(1) specifications is  $\text{D}^{\text{P}}$ -complete.*

385 **Proof.** For membership, observe that an input is a “yes” instance of EXACT-WI if and  
 386 only if it is a “yes” instance of WEAK IMPLEMENTATION *and* a “yes” instance of WEAK  
 387 IMPLEMENTATION COMPLEMENT (the problem where one asks whether  $\text{WI}(\mathcal{G}, \varphi, \beta) = \emptyset$ ).  
 388 Since the former problem is in NP and the latter problem is in coNP, membership in  $\text{D}^{\text{P}}$   
 389 follows. For the lower bound, we use the same reduction technique as in Theorem 16, and  
 390 reduce from EXACT TSP, a problem known to be  $\text{D}^{\text{P}}$ -hard [28, 29]. ◀

391 Following [27], we may naturally ask whether the optimal solution given by OPT-WI is  
 392 unique. We call this problem UOPT-WI. For some fixed budget  $\beta$ , it may be the case that  
 393 for two subsidy schemes  $\kappa, \kappa' \in \text{WI}(\mathcal{G}, \varphi, \beta)$  – we assume the implementation is efficient – we  
 394 have  $\kappa \neq \kappa'$  and  $\text{cost}(\kappa) = \text{cost}(\kappa')$ . With LTL specifications, it is not difficult to see that we  
 395 can solve UOPT-WI in polynomial space. Therefore, we have the following result.

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

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

- 398 1. Find the exact budget using binary search and WEAK IMPLEMENTATION as an oracle;
- 399 2. Use an NP oracle once to guess two distinct subsidy schemes with precisely this budget;
- 400 if no such subsidy schemes exist, return “yes”; otherwise, return “no”.

401 The above decision procedure clearly is in  $\Delta_2^{\text{P}}$  (for the upper bound). Furthermore, since  
 402 Theorem 16 implies  $\Delta_2^{\text{P}}$ -hardness [22] (for the lower bound), we have the following corollary.

403 ▶ **Corollary 19.** UOPT-WI with GR(1) specifications is  $\Delta_2^{\text{P}}$ -complete.

## 404 6.2 Optimality and Uniqueness in the Strong Domain

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

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

409 *What is the optimum budget  $\beta$  such that  $\text{SI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$ ?*

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

411 *Is  $b$  equal to the optimum budget for  $\text{SI}(\mathcal{G}, \varphi, \beta) \neq \emptyset$ ?*

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

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

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

416 For GR(1) specifications, observe that using the same arguments for the upper-bound  
 417 of OPT-WI with GR(1) specifications, we obtain the upper-bound for OPT-SI with GR(1)  
 418 specifications. Then, it follows that OPT-SI is in  $\text{FP}^{\Sigma_2^{\text{P}}}$ . For hardness, we define an  $\text{FP}^{\Sigma_2^{\text{P}}}$ -  
 419 complete problem, namely WEIGHTED MINQSAT<sub>2</sub>. Recall that in QSAT<sub>2</sub> we are given  
 420 a Boolean 3DNF formula  $\psi(\mathbf{x}, \mathbf{y})$  and sets  $\mathbf{x} = \{x_1, \dots, x_n\}, \mathbf{y} = \{y_1, \dots, y_m\}$ , with a set  
 421 of terms  $T = \{t_1, \dots, t_k\}$ . Define WEIGHTED MINQSAT<sub>2</sub> as follows. Given  $\psi(\mathbf{x}, \mathbf{y})$  and a  
 422 weight function  $c : \mathbf{x} \rightarrow \mathbb{Z}^{\geq}$ , WEIGHTED MINQSAT<sub>2</sub> is the problem of finding an assignment  
 423  $\vec{\mathbf{x}} \in \{0, 1\}^n$  with the least total weight such that  $\psi(\mathbf{x}, \mathbf{y})$  is true for every  $\vec{\mathbf{y}} \in \{0, 1\}^m$ . Observe  
 424 that WEIGHTED MINQSAT<sub>2</sub> generalises MINQSAT<sub>2</sub>, which is known to be  $\text{FP}^{\Sigma_2^{\text{P}}[\log n]}$ -hard  
 425 [12], *i.e.*, MINQSAT<sub>2</sub> is an instance of WEIGHTED MINQSAT<sub>2</sub>, where all weights are 1.

426 ▶ **Theorem 24.**  $\text{WEIGHTED MINQSAT}_2$  is  $\text{FP}^{\Sigma_2^P}$ -complete.

427 **Proof.** Membership follows from the upper-bound of  $\text{MINQSAT}_2$  [12]: since we have an  
 428 exponentially large input with respect to that of  $\text{MINQSAT}_2$ , by using binary search we will  
 429 need polynomially many calls to the  $\Sigma_2^P$  oracle. Hardness is immediate [12]. ◀

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

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

435 ▶ **Corollary 26.**  $\text{EXACT-SI}$  with  $\text{GR}(1)$  specifications is  $\text{D}_2^P$ -complete.

436 **Proof.** Membership follows from the fact that an input is a “yes” instance of  $\text{EXACT-SI}$  (with  
 437  $\text{GR}(1)$  specifications) if and only if it is a “yes” instance of  $\text{STRONG IMPLEMENTATION}$  and a  
 438 “yes” instance of  $\text{STRONG IMPLEMENTATION COMPLEMENT}$ , the decision problem where we  
 439 ask  $\text{SI}(\mathcal{G}, \varphi, \beta) = \emptyset$  instead. The lower bound follows from the hardness of  $\text{STRONG IMPLE-}$   
 440  $\text{MENTATION}$  and  $\text{STRONG IMPLEMENTATION COMPLEMENT}$  problems, which immediately  
 441 implies  $\text{D}_2^P$ -hardness [1, Lemma 3.2]. ◀

442 Furthermore, analogous to  $\text{UOPT-WI}$ , we also have the following corollaries.

443 ▶ **Corollary 27.**  $\text{UOPT-SI}$  with  $\text{LTL}$  specifications is  $\text{PSPACE}$ -complete.

444 ▶ **Corollary 28.**  $\text{UOPT-SI}$  with  $\text{GR}(1)$  specifications is  $\Delta_3^P$ -complete.

## 445 7 Conclusions & Related and Future Work

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

447 Although equilibrium design is closely related to mechanism design, as typically studied in  
 448 game theory [21], the two are not exactly the same. Two key features in mechanism design  
 449 are the following. Firstly, in a mechanism design problem, the designer is not given a game  
 450 structure, but instead is asked to provide one; in that sense, a mechanism design problem is  
 451 closer to a rational synthesis problem [14, 16]. Secondly, in a mechanism design problem, the  
 452 designer is only interested in the game’s outcome, which is given by the payoffs of the players  
 453 in the game; however, in equilibrium design, while the designer is interested in the payoffs of  
 454 the players as these may need to be perturbed by its budget, the designer is also interested –  
 455 and in fact primarily interested – in the satisfaction of a temporal logic goal specification,  
 456 which the players in the game do not take into consideration when choosing their individual  
 457 rational choices; in that sense, equilibrium design is closer to rational verification [17] than  
 458 to mechanism design. Thus, equilibrium design is a new computational problem that sits  
 459 somewhere in the middle between mechanism design and rational verification/synthesis.  
 460 Technically, in equilibrium design we go beyond rational synthesis and verification through  
 461 the additional design of subsidy schemes for incentivising behaviours in a concurrent and  
 462 multi-agent system, but we do not require such subsidy schemes to be incentive compatible  
 463 mechanisms, as in mechanism design theory, since the principal may want to reward only  
 464 a group of players in the game so that its temporal logic goal is satisfied, while rewarding  
 465 other players in the game in an unfair way – thus, leading to a game with a suboptimal  
 466 social welfare measure. In this sense, equilibrium design falls short with respect to the more  
 467 demanding social welfare requirements often found in mechanism design theory.

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

469 Typically, in rational synthesis and verification [14, 16, 17, 23] we want to check whether  
 470 a property is satisfied on some/every Nash equilibrium computation run of a reactive,  
 471 concurrent, and multi-agent system. These verification problems are primarily concerned  
 472 with qualitative properties of a system, while assuming rationality of system components.  
 473 However, little attention is paid to quantitative properties of the system. This drawback has  
 474 been recently identified and some work has been done to cope with questions where both  
 475 qualitative and quantitative concerns are considered [3, 6, 9, 10, 11, 18, 20, 34]. Equilibrium  
 476 design is new and different approach where this is also the case. More specifically, as  
 477 in a mechanism design problem, through the introduction of an external principal – the  
 478 designer in the equilibrium design problem – we can account for overall qualitative properties  
 479 of a system (the principal’s goal given by an LTL or a GR(1) specification) as well as for  
 480 quantitative concerns (optimality of solutions constrained by the budget to allocate additional  
 481 rewards/resources). Our framework also mixes qualitative and quantitative features in a  
 482 different way: while system components are only interested in maximising a quantitative  
 483 payoff, the designer is primarily concerned about the satisfaction of a qualitative (logic)  
 484 property of the system, and only secondarily about doing it in a quantitatively optimal way.

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

486 In recent years, there has been an interest in the analysis of rational outcomes of multi-agent  
 487 systems modelled as multi-player games. This has been done both with modelling and with  
 488 verification purposes. In those multi-agent settings, where AI agents can be represented as  
 489 players in a multi-player game, a focus of interest is on the analysis of (Nash) equilibria  
 490 in such games [8, 17]. However, it is often the case that the existence of Nash equilibria  
 491 in a multi-player game with temporal logic goals may not be guaranteed [16, 17]. For this  
 492 reason, there has been already some work on the introduction of desirable Nash equilibria in  
 493 multi-player games [2, 30]. This problem has been studied as a repair problem [2] in which  
 494 either the preferences of the players (given by winning conditions) or the actions available  
 495 in the game are modified; the latter one also being achieved with the use of normative  
 496 systems [30]. In equilibrium design, we do not directly modify the preferences of agents in the  
 497 system, since we do not alter their goals or choices in the game, but we indirectly influence  
 498 their rational behaviour by incentivising players to visit, or to avoid, certain states of the  
 499 overall system. We studied how to do this in an (individually) optimal way with respect to  
 500 the preferences of the principal in the equilibrium design problem. However, this may not  
 501 always be possible, for instance, because the principal’s temporal logic specification goal is  
 502 just not achievable, or because of constraints given by its limited budget.

503 **Future work: social welfare requirements and practical implementation.**

504 As discussed before, a key difference with mechanism design is that social welfare requirements  
 505 are not considered [25]. However, a benevolent principal might not see optimality as an  
 506 individual concern, and instead consider the welfare of the players in the design of a subsidy  
 507 scheme. In that case, concepts such as the *utilitarian social welfare* may be undesirable as the  
 508 social welfare maximising the payoff received by players might allocate all the budget to only  
 509 one player, and none to the others. A potentially better option is to improve fairness in the  
 510 allocation of the budget by maximising the *egalitarian social welfare*. Finally, given that the  
 511 complexity of equilibrium design is much better than that of rational synthesis/verification,  
 512 we should be able to have efficient implementations, for instance, as an extension of EVE [19].

513 — **References** —

- 514 1 Gadi Aleksandrowicz, Hana Chockler, Joseph Y. Halpern, and Alexander Ivrii. The compu-  
515 tational complexity of structure-based causality. *J. Artif. Int. Res.*, 58(1):431–451, January  
516 2017.
- 517 2 S. Almagor, G. Avni, and O. Kupferman. Repairing Multi-Player Games. In *CONCUR*,  
518 volume 42 of *LIPICs*, pages 325–339. Schloss Dagstuhl, 2015.
- 519 3 S. Almagor, O. Kupferman, and G. Perelli. Synthesis of controllable Nash equilibria in  
520 quantitative objective games. In *IJCAI*, pages 35–41, 2018.
- 521 4 B. Aminof, V. Malvone, A. Murano, and S. Rubin. Graded Strategy Logic: Reasoning about  
522 Uniqueness of Nash Equilibria. In *AAMAS*, pages 698–706. ACM, 2016.
- 523 5 R. Bloem, B. Jobstmann, N. Piterman, A. Pnueli, and Y. Sa’ar. Synthesis of reactive(1)  
524 designs. *Journal of Computer and System Sciences*, 78(3):911–938, 2012.
- 525 6 A. Bohy, V. Bruyère, E. Filiot, and J. Raskin. Synthesis from LTL Specifications with  
526 Mean-Payoff Objectives. In *TACAS*, pages 169–184, 2013.
- 527 7 U. Boker, K. Chatterjee, T. A. Henzinger, and O. Kupferman. Temporal Specifications with  
528 Accumulative Values. *ACM Transactions on Computational Logic*, 15(4):27:1–27:25, 2014.
- 529 8 P. Bouyer, R. Brenguier, N. Markey, and M. Ummels. Pure Nash Equilibria in Concurrent  
530 Deterministic Games. *Logical Methods in Computer Science*, 11(2), 2015.
- 531 9 K. Chatterjee and L. Doyen. Energy parity games. *Theoretical Computer Science*, 458:49–60,  
532 2012.
- 533 10 K. Chatterjee, L. Doyen, T. Henzinger, and J. Raskin. Generalized Mean-payoff and Energy  
534 Games. In *FSTTCS*, pages 505–516, 2010.
- 535 11 K. Chatterjee, T. A. Henzinger, and M. Jurdzinski. Mean-Payoff Parity Games. In *LICS*,  
536 pages 178–187. IEEE Computer Society, 2005.
- 537 12 H. Chockler and J. Halpern. Responsibility and Blame: A Structural-Model Approach. *Journal*  
538 *of Artificial Intelligence Research*, 22:93–115, 2004.
- 539 13 Aldric Degorre, Laurent Doyen, Raffaella Gentilini, Jean-François Raskin, and Szymon Tor-  
540 uńczyk. Energy and mean-payoff games with imperfect information. In Anuj Dawar and  
541 Helmut Veith, editors, *Computer Science Logic*, pages 260–274, Berlin, Heidelberg, 2010.  
542 Springer Berlin Heidelberg.
- 543 14 D. Fisman, O. Kupferman, and Y. Lustig. Rational Synthesis. In *TACAS*, volume 6015 of  
544 *LNCS*, pages 190–204. Springer, 2010.
- 545 15 J. Gutierrez, P. Harrenstein, G. Perelli, and M. Wooldridge. Nash Equilibrium and Bisimulation  
546 Invariance. In *CONCUR*, volume 85 of *LIPICs*, pages 17:1–17:16. Schloss Dagstuhl, 2017.
- 547 16 J. Gutierrez, P. Harrenstein, and M. Wooldridge. Iterated Boolean Games. *Information and*  
548 *Computation*, 242:53–79, 2015.
- 549 17 J. Gutierrez, P. Harrenstein, and M. Wooldridge. From Model Checking to Equilibrium  
550 Checking: Reactive Modules for Rational Verification. *Artificial Intelligence*, 248:123–157,  
551 2017.
- 552 18 J. Gutierrez, A. Murano, G. Perelli, S. Rubin, and M. Wooldridge. Nash Equilibria in  
553 Concurrent Games with Lexicographic Preferences. In *IJCAI*, pages 1067–1073, 2017.
- 554 19 J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. EVE: A Tool for Temporal Equilibrium  
555 Analysis. In *ATVA*, volume 11138 of *LNCS*, pages 551–557. Springer, 2018.
- 556 20 J. Gutierrez, M. Najib, G. Perelli, and M. Wooldridge. On Computational Tractability for  
557 Rational Verification. In *IJCAI*, 2019. To appear.
- 558 21 L. Hurwicz and S. Reiter. *Designing Economic Mechanisms*. Cambridge University Press,  
559 2006.
- 560 22 M. Krentel. The Complexity of Optimization Problems. *Journal of Computer and System*  
561 *Sciences*, 36(3):490 – 509, 1988.
- 562 23 O. Kupferman, G. Perelli, and M. Y. Vardi. Synthesis with rational environments. *Annals of*  
563 *Mathematics and Artificial Intelligence*, 78(1):3–20, 2016.

## 18:16 Equilibrium Design for Concurrent Games

- 564 **24** M. Wooldridge and U. Endriss and S. Kraus and J. Lang. Incentive engineering for Boolean  
565 games. *Artificial Intelligence*, 195:418 – 439, 2013.
- 566 **25** M. Maschler, E. Solan, and S. Zamir. *Game Theory*. Cambridge University Press, 2013.
- 567 **26** M.J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, 1994.
- 568 **27** C. Papadimitriou. On the Complexity of Unique Solutions. *Journal of the ACM*, 31(2):392–400,  
569 1984.
- 570 **28** C. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
- 571 **29** C. Papadimitriou and M. Yannakakis. The complexity of facets (and some facets of complexity).  
572 *Journal of Computer and System Sciences*, 28(2):244 – 259, 1984.
- 573 **30** G. Perelli. Enforcing equilibria in multi-agent systems. In *Proceedings of the 18th International*  
574 *Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 188–196,  
575 2019.
- 576 **31** A. Pnueli. The Temporal Logic of Programs. In *FOCS*, pages 46–57. IEEE, 1977.
- 577 **32** A. Pnueli and R. Rosner. On the Synthesis of a Reactive Module. In *POPL*, pages 179–190.  
578 ACM Press, 1989.
- 579 **33** M. Ummels and D. Wojtczak. The Complexity of Nash Equilibria in Limit-Average Games.  
580 In *CONCUR*, pages 482–496, 2011.
- 581 **34** Y. Velner, K. Chatterjee, L. Doyen, T. Henzinger, A. Rabinovich, and J. Raskin. The  
582 Complexity of Multi-Mean-Payoff and Multi-Energy Games. *Information and Computation*,  
583 241:177–196, 2015.
- 584 **35** U. Zwick and M. Paterson. The Complexity of Mean Payoff Games on Graphs. *Theoretical*  
585 *Computer Science*, 158(1):343 – 359, 1996.