Proceedings of the 21st International Conference on Principles of Knowledge Representation and Reasoning
Main Track

Incentive Design for Rational Agents

David Hyland[1], **Munyque Mittelmann**[2], **Aniello Murano**[2], **Giuseppe Perelli**[3], **Michael Wooldridge**[1]
[1]University of Oxford
[2]University of Naples Federico II
[3]Sapienza University of Rome
{david.hyland, mjw}@cs.ox.ac.uk, {munyque.mittelmann, aniello.murano}@unina.it,
perelli@di.uniroma1.it

Abstract

We introduce *Incentive Design*: a new class of problems for equilibrium verification in multi-agent systems. In our model, agents attempt to maximize their utility functions, which are expressed as formulae in $\mathsf{LTL}[\mathcal{F}]$, a quantitative extension of Linear Temporal Logic with functions computable in polynomial time. We assume agents are rational, in the sense that they adopt strategies consistent with game theoretic solution concepts such as Nash equilibrium. For each solution concept we consider, we analyze the problems of verifying whether an incentive scheme achieves a societal objective and finding one that does so, whether it be social welfare or any other aggregate measure of collective well-being. We study both static and dynamic incentive schemes, showing that the latter are more powerful than the former. Finally, we solve the incentive verification and synthesis problems for all the solution concepts we consider, and analyze their complexity.

1 Introduction

The design of mechanisms for aggregating preferences in multi-agent systems is a key problem in algorithmic game theory (Nisan et al. 2007). One classic approach is to design a system such that it satisfies the designer's objective (e.g., a property such as efficiency) *under the assumption that agents act rationally*. In traditional approaches to formal verification, one is interested in determining whether a given property, expressed as a temporal logic formula, holds on some or all of the possible executions of a system (Baier and Katoen 2008). However, when components of the system are *rational agents*, we can instead focus our attention on only those outcomes of the system that are rational from the perspective of the agents. This is the central motivation behind the fields of rational verification and rational synthesis (Condurache et al. 2016; Filiot, Gentilini, and Raskin 2018; Almagor, Kupferman, and Perelli 2018; Fisman, Kupferman, and Lustig 2010; Kupferman and Shenwald 2022; Brice, Raskin, and van den Bogaard 2023).

In many settings, the underlying structure that describes the agents' interactions with the environment and other agents is already known. While those systems may not comply with the designer's objective, a complete redesign is not always feasible. For instance, a country may have environmental legislation but fail to achieve its sustainability objectives because the incentives of private individuals and companies are not aligned with these higher-level objectives. In such cases, policymakers may, for example, resort to establishing taxes based on a company's pollution rate, or subsidizing public transportation fees to incentivize its use.

In this paper, we are interested in designing incentives in multi-agent systems to motivate rational agents to achieve the designer's objective. To achieve this, we introduce a new class of problems for equilibrium verification that we call *Incentive Design*. As usual in game theory, we consider agents who seek to maximize their personal utility function. In our setting, such utilities are expressed as formulae of $\mathsf{LTL}[\mathcal{F}]$ (Almagor, Boker, and Kupferman 2016), a quantitative extension of Linear Temporal Logic (Pnueli 1977) with functions that are computable in polynomial time. This enables the representation of agents' utilities as *quantitative temporal specifications*. The rationality of an outcome is defined with respect to game theoretic solution concepts, including *dominant strategy equilibrium*, *Nash equilibrium*, *immune*, and *resilient equilibria*. For each of these solution concepts, we study the problem of finding and verifying an appropriate *incentive mechanism* to achieve a societal objective, be it the social welfare or any other aggregating measure of collective wellbeing. To address these challenges, we introduce two problems: incentive verification and incentive synthesis. In incentive *verification*, the aim is to verify whether the application of a given incentive scheme guarantees some level of satisfaction for a given objective. In incentive *synthesis*, we are interested in finding an incentive scheme, if it exists, which guarantees that the satisfaction value of some objective attains at least a certain level.

We focus on two major classes of incentive mechanisms: *static* and *dynamic*. The former assigns new satisfaction values to some of the variables in the system, regardless of the execution history. The latter dynamically reassigns satisfaction values based on the history of the game thus far. We demonstrate that dynamic incentives are more powerful than their static counterpart. For all solution concepts that we consider, we show that incentive verification is 2EXPTIME-complete for both static and dynamic incentives, and is hence no harder than the corresponding qualitative rational verification problems. Finally, we solve the incentive synthesis problem and prove that it is 2EXPTIME-complete in the static case and in 3EXPTIME in the dynamic case.

Related work. Our work is inspired by a rich line of research analyzing and designing multi-agent systems composed of rational agents, known as rational verification and synthesis.

Also related is the problem of repairing the goals of players (Almagor, Avni, and Kupferman 2015) and synthesis of dynamic norms (Alechina et al. 2022). This problem consists at modifying the objectives or strategies of the players in order to obtain stability or to improve social welfare. Recent work has also proposed the application of formal methods to the verification and synthesis of mechanisms for social choice (Maubert et al. 2021; Mittelmann et al. 2022; Belardinelli et al. 2022; Narayanasamy 2022).

The problem of designing incentives has been considered from the perspective of economics, control theory, and machine learning (Ratliff et al. 2019). The economics approach typically focuses on the treatment of information asymmetries and the design of incentive-compatible mechanisms (Parkes et al. 2010; Laffont and Martimort 2009). The control theory approach introduces variables to encode information about the evolution of the environment, as it depends on agents' choices (Olsder 2009; Weber 2011). Learning approaches to dynamic incentives in multi-agent settings focus solely on adaptively modifying the rewards for quantitative reward-maximising agents (Centeno and Billhardt 2011; Ratliff and Fiez 2020; Yang et al. 2022). Differently from this line of work, our approach is rooted in formal methods, which guarantees the correctness of incentive schemes with respect to the input specification. Instead of specifying a particular global objective (such as utilitarian social welfare maximisation) and then designing a solution to achieve it in a wide variety of environments, our approach is more general in that we allow arbitrary global objectives specified in $\mathsf{LTL}[\mathcal{F}]$ to be provided as input to our procedures.

The closest work to ours is a line of research on *Incentive Engineering*. In this direction, (Wooldridge et al. 2013) first considered taxation schemes for one-shot Boolean games, an approach that was later extended to concurrent games where players have lexicographic LTL and mean-payoff objectives (Hyland, Gutierrez, and Wooldridge 2023b). A variant of the problem, where the incentive designer can only observe some subset of the system variables, was studied by (Hyland, Gutierrez, and Wooldridge 2023a). These works investigate how to incentivize agents by introducing costs to their actions. Our notion of incentives is more general, as we allow for changes to any of the atomic features of the game states. For the first time, we consider the setting in which both the agents and the incentive designer have goals expressed in a quantitative temporal logic.

2 Preliminaries

For the remainder of the paper, we fix a set of atomic propositions AP, a set of agents Ag, and a set of strategy variables Var. We also let $\mathcal{F} \subseteq \{f : [-1,1]^m \to [-1,1] \mid m \in \mathbb{N}\}$ be a set of functions computable in polynomial time over $[-1,1]$ of possibly different arities, that will parameterize the logics we consider. With slight abuse of notation, we denote by $f \in \mathcal{F}$ both the function and the corresponding function symbol. It will be clear from the context what the symbol corresponds to.

In this paper, we consider a variant of the classic model of concurrent game structures (CGS) (Alur, Henzinger, and Kupferman 2002), where atomic propositions are associated with a weight in different states:

Definition 1. A *weighted concurrent game structure* (wCGS) is a tuple $G = (\mathrm{AP}, \mathrm{Ag}, (\mathrm{Ac}_a)_{a \in \mathrm{Ag}}, V, v_\iota, \delta, \ell)$ where (i) AP is a finite set of *atomic propositions*; (ii) $\mathrm{Ag} = \{1, ..., n\}$ is a finite set of n *agents*; (iii) Ac_a is a finite set of *actions* for agent a and $\mathrm{Ac} = \bigtimes_{a \in \mathrm{Ag}} \mathrm{Ac}_a$ is the set of *joint actions*; (iv) V is a finite set of *positions*; (v) $v_\iota \in V$ is an *initial position*; (vi) $\delta : V \times \bigtimes_{a \in \mathrm{Ag}} \mathrm{Ac}_a \to V$ is a *transition function*; (vii) $\ell : V \times \mathrm{AP} \to [-1, 1]$ is a *weight function*.

In a position $v \in V$, each player a chooses an action $c_a \in \mathrm{Ac}_a$, and the state proceeds to the position $\delta(v, \boldsymbol{c})$ where $\boldsymbol{c}$ is an *action profile* $(c_a)_{a \in \mathrm{Ag}}$.

Given a coalition of agents $\mathrm{C} \subseteq \mathrm{Ag}$, we write $\boldsymbol{o}_\mathrm{C}$ for a tuple of objects $(o_a)_{a \in \mathrm{C}}$, one for each agent in C, and such tuples are called *profiles*. We also write $\boldsymbol{o}$ for $\boldsymbol{o}_{\mathrm{Ag}}$. Given a profile $\boldsymbol{o}$ and $a \in \mathrm{Ag}$, we let o_a be agent a's component, and $\boldsymbol{o}_{-a}$ is used to denote the profile $(o_b)_{b \neq a}$.

A *play* $\pi = v_1 v_2 ...$ is an infinite sequence of positions such that for every $i \geq 1$ there exists an action profile $\boldsymbol{c}$ such that $\delta(v_i, \boldsymbol{c}) = v_{i+1}$. We write $\pi_i = v_i$ for the position at index i in play π and $\pi_{\geq i} = v_i v_{i+1} ...$ for the suffix of π starting from the position at index i. A *history* h is a finite prefix of a play, last(h) is the last position of history h, $|h|$ is the length of h and Hist is the set of histories.

We will find it useful to generalise the weight function so that it may vary over the course of a play. Given a set AP of atomic propositions, an infinite sequence of weight functions $\vec{\ell} = \ell_1 \ell_2 ...$, and a play π, the $(\mathrm{AP}, \vec{\ell})$*-labelling* of π, written $L(\pi; \mathrm{AP}, \vec{\ell})$, is given by the infinite sequence of weight assignments

$$L(\pi; \mathrm{AP}, \vec{\ell}) := (\ell_1(v_1, p))_{p \in \mathrm{AP}} (\ell_2(v_2, p))_{p \in \mathrm{AP}} ...$$

that is obtained by evaluating each weight function ℓ_i in the sequence $\vec{\ell}$ with at the corresponding position v_i over the set AP. In a standard wCGS, the sequence $\vec{\ell}$ is given by the repeating sequence $\vec{\ell} = \ell\ell\ell....$ Later, we explore the possibility for an incentive designer to dynamically modify the weight functions, leading to a non-trivial sequence $\vec{\ell}$.

A *strategy* for an agent a is a function $\sigma_a : \mathrm{Hist} \to \mathrm{Ac}_{\mathrm{Ag}}$ that maps each history to an action. We let Str_a be the set of strategies for a and let $\mathit{Str} = \prod_{a \in \mathrm{Ag}} \mathit{Str}_a$ be the set of strategy profiles. An *assignment* $\mathcal{A} : \mathrm{Ag} \cup \mathrm{Var} \to \mathit{Str}$ is a function from players and variables to strategies. For an assignment $\mathcal{A}$, an agent a, and a strategy σ for a, $\mathcal{A}[a \mapsto \sigma]$ is the assignment that maps a to σ and is otherwise equal to $\mathcal{A}$, and $\mathcal{A}[s \mapsto \sigma]$ is defined similarly, where s is a variable. For a strategy profile $\boldsymbol{\sigma} = (\sigma_a)_{a \in \mathrm{Ag}}$, we write $\mathcal{A}[\mathrm{Ag} \mapsto \boldsymbol{\sigma}]$ for the assignment where $\mathcal{A}[a \mapsto \sigma_a]$ for each $a \in \mathrm{Ag}$.

For an assignment $\mathcal{A}$ and a history h, we let $\mathrm{Out}(\mathcal{A}, h)$ be the unique play that continues h following the strategies assigned by $\mathcal{A}$. Formally, $\mathrm{Out}(\mathcal{A}, h)$ is the play $h v_0 v_1 ...$ such that for all $i \geq 0$, $v_i = \delta(v_{i-1}, \boldsymbol{c})$ where for all $a \in \text{Ag}$, $\boldsymbol{c}_a = \mathcal{A}(a)(hv_0...v_{i-1})$, and $v_{-1} = \text{last}(h)$.

Example 1. Consider a scenario involving two manufacturing firms who share the usage of a river and a village community who live downstream of the firms on the same river. We represent this by the set $\text{Ag} = \{1, 2, 3\}$, where agents 1 and 2 are the firms. At every time step, each firm has two choices: to discharge waste water directly into the river, or to first treat the waste water to clean it before discharging it into the river. We represent this by letting $\text{Ac}_i = \{c_a, d_a\}, a \in \{1, 2\}$ to denote the 'clean' and 'discharge' actions for the two firms respectively. The c_a action incurs a fixed cost for firm a, whereas the d_a action incurs no immediate cost for the firm. At every time step, the village community also has two choices: to use the river water for its day-to-day activities, or to use another water source. We denote this by the actions $\text{Ac}_3 = \{r, o\}$, which stand for using the 'river' or an 'other' source of water respectively. The use of the river by the village incurs a cost proportional to the level of contamination of the river, up to a threshold of k units, in which case the river becomes unusable and incurs a very high cost. The use of the other source of water incurs a high but fixed cost for the community.

In this setting, suppose that the river carries water away (whether clean or dirty) at a constant rate of 1 unit per timestep. We can define states to represent varying degrees of contamination of the river water. Let $V = \{v_0\} \cup \{v_i^{\boldsymbol{c}} : i \in \{0, ..., k\}, \boldsymbol{c} \in \text{Ac}\}$, where $k \in \mathbb{N}_+$ is a threshold such that if the river contamination level ever reaches k units, the river ecosystem is destroyed and the water becomes permanently unusable. The duplication of states for each joint action is used to record the previously taken joint action. The river starts in a clean state ($v_\iota = \{v_0\}$), and the transition function between intermediate states $v_i^{\boldsymbol{c}}, i \in \{1, ..., k-1\}$ can be represented by $\delta(v_i^{\boldsymbol{c}}, \boldsymbol{c}') = v_j^{\boldsymbol{c}'}$, where $j = \max(0, \min(k, i + (d - c)/2))$ and D represents the number of agents who selected the d action in $\boldsymbol{c}$. Transitions out of the initial state are defined in the same manner, and for the ruin states $v_k^{\boldsymbol{c}}$, we let $\delta(v_k^{\boldsymbol{c}}, \boldsymbol{c}') = v_k^{\boldsymbol{c}'}$. In other words, if both firms choose the discharge action, the water quality deteriorates by 1 unit, up to the point of ruin. If only one firm chooses the discharge action, the overall quality remains as it is, and if both firms choose the clean action, the river quality improves by 1 unit, up to a maximum level.

We use an atomic proposition q to represent water quality. We can assign $\ell(v_i^{\boldsymbol{c}}, q) = 1 - 2i/k$ for all $\boldsymbol{c} \in \text{Ac}$, so that the water quality is 1 when the river is perfectly clean and -1 when the river is unusable.

In order to model the effects of agent actions on their objectives, we additionally assume that the game has atomic propositions $\{u_1, u_2, u_3\}$, which represent the 'utility' that each player achieves at a particular state. For a firm $a \in \{1, 2\}$, the weight function for these variables at a state $v_i^{\boldsymbol{c}}$ is given by $\ell(v_i^{\boldsymbol{c}}, u_a) = 0.2$ if $\boldsymbol{c}_a = c_a$ and $\ell(v_i^{\boldsymbol{c}}, u_a) = 0.4$ if $\boldsymbol{c}_a = d_a$. For the community, we let $\ell(v_i^{\boldsymbol{c}}, u_3) = \ell(v_i^{\boldsymbol{c}}, q)$ if $\boldsymbol{c}_3 = r$, $\ell(v_i^{\boldsymbol{c}}, u_3) = 0$ if $\boldsymbol{c}_3 = o$. Finally, we let $\ell(v_0, q) = 1$ and $\ell(v_0, p) = 0$ for all $p \in \text{AP} \setminus \{q\}$.

Quantitative Linear Temporal Logic. To specify players' objectives and define the concept of a rational outcome, we make use of Linear Temporal Logic with quantitative semantics ($\mathsf{LTL}[\mathcal{F}]$) (Almagor, Boker, and Kupferman 2016). We work with this version over LTL primarily because assuming that formula satisfaction values (over which utilities are typically defined) only take on the values of 0 or 1 limits the kinds of incentive schemes that an incentive designer can devise and implement. With quantitative logics such as $\mathsf{LTL}[\mathcal{F}]$, there is much more flexibility in capturing the more common conception of an incentive. A classic example of this is in income taxes where the utility of employees depends on their net income, which cannot be reduced to a Boolean outcome.

Definition 2. The syntax of $\mathsf{LTL}[\mathcal{F}]$ is defined by the BNF

$$\varphi ::= p \mid f[\varphi, ..., \varphi] \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

where $p \in \text{AP}$ and $f \in \mathcal{F}$.

Here, $\mathbf{X}$ and $\mathbf{U}$ are the usual temporal operators "next" and "until". The meaning of $f[\varphi_1, ..., \varphi_n]$ depends on the function f. We use $\top$, $\vee$, and $\neg$ to denote, respectively, function 1, function $x, y \mapsto \max(x, y)$ and function $x \mapsto -x$.

We assume here that an agent a's decision-making aims to maximise the satisfaction value of an $\mathsf{LTL}[\mathcal{F}]$ formula γ_a. Once goals for each agent are specified, we have a *weighted concurrent game*, defined formally as follows:

Definition 3. Let $G = (\text{AP}, \text{Ag}, (\text{Ac}_a)_{a \in \text{Ag}}, V, \delta, \ell, V_\iota)$ be a wCGS, and $\mathcal{A}$ an assignment. The satisfaction value $[\![\varphi]\!]_{\mathcal{A}}^G(h) \in [-1, 1]$ of an $\mathsf{LTL}[\mathcal{F}]$ formula φ in a history h is defined as follows, where π denotes $\text{Out}(\mathcal{A}, h)$:

$$\begin{aligned}
[\![p]\!]_{\mathcal{A}}^G(h) &= \ell(\text{last}(h), p) \\
[\![f[\varphi_1, ..., \varphi_m]]\!]_{\mathcal{A}}^G(h) &= f([\![\varphi_1]\!]_{\mathcal{A}}^G(h), ..., [\![\varphi_m]\!]_{\mathcal{A}}^G(h)) \\
[\![\mathbf{X}\varphi]\!]_{\mathcal{A}}^G(h) &= [\![\varphi]\!]_{\mathcal{A}}^G(\pi_{|h|+1}) \\
[\![\varphi_1\mathbf{U}\varphi_2]\!]_{\mathcal{A}}^G(h) &= \sup_{i \geq 0} \min\big([\![\varphi_2]\!]_{\mathcal{A}}^G(\pi_{|h|+i}), \\
&\qquad \min_{0 \leq j < i} [\![\varphi_1]\!]_{\mathcal{A}}^G(\pi_{|h|+j})\big)
\end{aligned}$$

When the satisfaction value does not depend on the assignment, we write $[\![\varphi]\!]^G(h)$ for $[\![\varphi]\!]_{\mathcal{A}}^G(h)$ where $\mathcal{A}$ is any assignment. Additionally, because arena transitions are deterministic with respect to actions in this setting, every assignment gives rise to a unique play. Therefore, we also define the semantics of $\mathsf{LTL}[\mathcal{F}]$ formulae over a play π with respect to a valuation function ℓ over a set AP, written $[\![\varphi]\!]_\pi^{\text{AP}, \ell}$, in the same manner as above.

We define some abbreviations, corresponding to classical logic and LTL operators: $\bot := \neg\top$, $\varphi \wedge \varphi' := \neg(\neg\varphi \vee \neg\varphi')$, $\varphi \to \varphi' := \neg\varphi \vee \varphi'$, $\mathbf{F}\psi := \top\mathbf{U}\psi$, $\mathbf{G}\psi := \neg\mathbf{F}\neg\psi$.

We assume that $\mathcal{F}$ contains the comparison function

$$\leq : (x, y) \mapsto \begin{cases} 1 & \text{if } x \leq y, \\ -1 & \text{otherwise,} \end{cases}$$

similarly, the equality function is defined as

$$= : (x, y) \mapsto \begin{cases} 1 & \text{if } x = y, \\ -1 & \text{otherwise} \end{cases}$$

Definition 4. A *weighted concurrent game* (wCG) is a tuple $\mathcal{G} = (G, (\gamma_a)_{a\in \text{Ag}})$, where G is a wCGS and γ_a is an $\mathsf{LTL}[\mathcal{F}]$ formula over AP, for all $a \in$ Ag.

Solution concepts. In this paper we consider agents with $\mathsf{LTL}[\mathcal{F}]$ goals, where the formula γ_a denotes the goal of agent a. The utility of agent a in a wCGS G given the strategy profile $\boldsymbol{\sigma} = (\sigma_a)_{a\in\text{Ag}}$ is denoted $\text{util}^G_a(\boldsymbol{\sigma})$ and is defined as the satisfaction value of γ_a in the initial state of G when agents are assigned to their strategies in $\boldsymbol{\sigma}$, that is, $\text{util}^G_a(\boldsymbol{\sigma}) = [\![\gamma_a]\!]^G_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}(v_\iota)$.

We now recall some key game theoretic solution concepts. Let $\mathcal{G} = (G, (\gamma_a)_{a\in\text{Ag}})$ be a wCG and $\boldsymbol{\sigma} = (\sigma_a)_{a\in\text{Ag}}$ be a strategy profile. We say that σ is a dominant strategy equilibrium if the strategy associated with each agent weakly maximizes her utility, for all possible strategies of other agents. Formally, $\boldsymbol{\sigma} = (\sigma_a)_{a\in\text{Ag}}$ is a *dominant strategy equilibrium* in $\mathcal{G}$ if for all agent a and for all strategy profiles $\boldsymbol{\sigma'} \in \prod_{a\in\text{Ag}} \mathit{Str}_a$, $\text{util}^G_a(\boldsymbol{\sigma'}) \le \text{util}^G_a(\boldsymbol{\sigma'}_{-a}, \sigma_a)$.

We say σ is a Nash equlibrium if no agent can increase her utility with a unilateral change of strategy. Formally, $\boldsymbol{\sigma} = (\sigma_a)_{a\in\text{Ag}}$ is a *Nash equilibrium* in $\mathcal{G}$ if for all agents a, and all strategy $\sigma'_a \in \mathit{Str}_a$, we have that $\text{util}^G_a(\boldsymbol{\sigma}_{-a}, \sigma') \le \text{util}^G_a(\boldsymbol{\sigma})$.

In an m-resilient equilibrium, we consider deviations by coalitions of agents rather than individuals: this solution concept tolerates deviations of up to m agents (Abraham, Dolev, and Halpern 2008). Formally, $\boldsymbol{\sigma} = (\sigma_a)_{a\in\text{Ag}}$ is an *m-resilient equilibrium* in $\mathcal{G}$ if for all coalitions C $\subseteq$ Ag with size at most m, and all partial strategy profiles $\boldsymbol{\sigma'}_{\text{Ag}\setminus\text{C}} \in \prod_{a\in\text{Ag}\setminus\text{C}} \sigma_a$, we have that $\text{util}^G_a(\boldsymbol{\sigma}_{\text{Ag}\setminus\text{C}}, \boldsymbol{\sigma'}_{\text{C}}) \le \text{util}^G_a(\boldsymbol{\sigma})$.

We write $\text{DSE}(\mathcal{G}), \text{NE}(\mathcal{G}), \text{RE}_m(\mathcal{G})$ to denote the sets of dominant strategy equilibria, Nash equilibria, and m-resilient equilibria of a game $\mathcal{G}$, respectively.

Example 2. Continuing from Example 1, we assume that the objective of all players is to maintain a high level of utility at all times. Suppose in addition that there is a regulator who is able to impose taxes on the firms. We can model this by introducing two new variables t_1, t_2 to represent the taxes imposed on firms 1 and 2 respectively. Taxes are initially set to 0, i.e., $\ell(v, t_a) = 0$ for all $v \in V, a \in \{1, 2\}$. Given this, we define the goals for the players on a given play π to be $\gamma_a = \mathbf{G}(u_a - t_a)$ for $a \in \{1, 2\}$ and $\gamma_3 = \mathbf{G}u_3$, where the '$-$' function is defined in the obvious manner.

In this scenario, the two firms do not internalize the effect of their actions on the quality of the river. Since discharging wastewater directly into the river is cheaper for them, there is a Nash equilibrium where both firms choose the d action every round, leading to the spoiling of the river after k time steps. An incentive designer may wish to avoid this scenario by putting in place appropriate incentives to discourage indiscriminate pollution. To realize this aim, we now define our model of incentives.

3 Incentives

Given a wCG $\mathcal{G} = (\text{AP}, \text{Ag}, (\text{Ac}_a)_{a\in\text{Ag}}, V, v_\iota, \delta, \ell, (\gamma_a)_{a\in\text{Ag}})$, we interpret incentives as assignments of satisfaction values to a particular set of propositional variables.

Let $U \subseteq$ AP be a set of propositional variables in $\mathcal{G}$. An *incentive scheme* over U is a weight function $\theta : U \to [-1, 1]$ which induces a modified weight function ℓ_θ such that for all $v \in V$ and $p \in$ AP, we have

$$\ell_\theta(p) = \begin{cases} \theta(p) & p \in U \\ \ell(v, p) & p \notin U. \end{cases}$$

Defined in this way, an incentive scheme applied at a particular point in time modifies the satisfaction values of the variables in U. Depending on the interpretation of the variables, this definition of incentives may take on different meanings. For example, if a variable p represents the monetary cost of being in a particular state, then an incentive on p may be interpreted as a tax or subsidy on the visitation of a state. Moreover, the variables can be used to represent incentives that are agent-specific or ones that are common to multiple agents, depending on the dependencies of utility functions on these variables. Thus, the set U contains the possible variables in the players' shared environment, which the regulator can intervene upon to influence their decision-making, and are referred to as *incentivized variables*.

Let Θ denote the set of incentive schemes over a set $U \subseteq$ AP. We will assume that incentive schemes have a fixed level of granularity. More specifically, we assume that Θ is a finite set of incentive schemes, and we say that an incentive scheme θ has a granularity $g \in \mathbb{Q}^+$ if for all $v \in V, p \in U$, we have $\theta(p) - \ell(v, p) = k \cdot g$, for some $k \in \mathbb{Z}$.

A *dynamic incentive scheme* is a function T : Hist $\to \Theta$ that maps each history to an incentive scheme that is applied on the same round, i.e., the incentive scheme $\theta_k = T(v_1...v_k)$ is applied in round k of the game. A dynamic incentive scheme thus defines a dynamic modification to the satisfaction values of a fixed subset of the propositional variables AP defined in a wCG $\mathcal{G}$. In contrast, a *static incentive scheme* is a dynamic incentive scheme that outputs the same incentive scheme for all histories ending with the same state. In other words, a static incentive scheme is a *memoryless* dynamic one, which outputs the same incentive at a given state, no matter what history led to the arrival at that state.

It is relatively straightforward to interpret what the application of a static incentive scheme T over a set U means in a given wCG, and we write $\mathcal{G}_T := \mathcal{G}_{\theta^T} = (\text{AP}, \text{Ag}, (\text{Ac}_a)_{a\in\text{Ag}}, V, v_\iota, \delta, \ell_{\theta^T}, (\gamma_a)_{a\in\text{Ag}})$, where θ^T is the incentive scheme that T outputs at each point in the game. A dynamic incentive scheme, on the other hand, can be thought of as inducing a dynamic (history-dependent) weight function on the game $\mathcal{G}$. In particular, given a play π of a game $\mathcal{G}$, a dynamic incentive scheme T induces an infinite sequence $\vec{\ell}_T(\pi) = \ell_{\theta^1}\ell_{\theta^2}...$ of weight functions, such that for every history $h = v_1...v_k$ of π, we have $\theta^k = T(h)$.

More generally, dynamic weight functions be accommodated in our model by defining new semantics for the satisfaction value $[\![\varphi]\!]^{G,\vec{\ell}}_{\mathcal{A}}(h)$ of an $\mathsf{LTL}[\mathcal{F}]$ formula in a wCGS G, given an infinite sequence of weight functions $\vec{\ell} = \ell_1\ell_2...$ over a finite set $\mathcal{L} = \{\ell_1, \ell_2, ..., \ell_{|\mathcal{L}|}\}$ of possible weight functions. This is achieved by letting

$$[\![p]\!]^{G,\vec{\ell}}_{\mathcal{A}}(h) := \ell_{|h|}(\text{last}(h), p),$$

while the semantics for f, the $\mathbf{X}$ operator, and the $\mathbf{U}$ operator remain the same. Thus, in the specific context of a dynamic incentive scheme T over a set $U \subseteq$ AP, we then have

$$[\![p]\!]_{\mathcal{A}}^{G_T}(h) := [\![p]\!]_{\mathcal{A}}^{G,\vec{\ell}_T}(h) = \begin{cases} T(h)(p) & p \in U \\ \ell(\text{last}(h), p) & p \notin U. \end{cases}$$

We also define the semantics of $\mathsf{LTL}[\mathcal{F}]$ over $(\text{AP}, \vec{\ell})$-labelled plays $L(\pi; \text{AP}, \vec{\ell})$, written $[\![\varphi]\!]_{\pi}^{\text{AP},\vec{\ell}}$, in the same way.

Due to the fixed granularity assumption, a dynamic incentive scheme's set of possible outputs is always finite. As we demonstrate later, this assumption is important as it allows one to model check $\mathsf{SL}[\mathcal{F}]$ formulae with this modified semantics, albeit at some additional computational cost. However, this does not affect the worst-case complexity results we derive here.

We argue that for most practical purposes, this is a reasonable assumption to make, as it will usually be the case that beyond a certain level of granularity, the benefits of introducing an even finer partition of the incentive space will be outweighed by the additional complexity introduced in computing the optimal incentive scheme and then implementing it in practice. Examples of granularity appearing in real-world incentive policies include taxation rates or the US Federal Reserve's interest rate on reserve balances.

Example 3. With incentives defined, we can now consider possible incentive implementations by a regulator in our running example. In most instances, a regulator typically has an *objective* that they would like to see accomplished as the result of the implementation of incentive schemes.

In our case, suppose that the regulator wants to impose taxes on the firms to maintain a stable and acceptable water quality level. To achieve this aim, the regulator can intervene on the t_a variables (i.e., $U = \{t_1, t_2\}$), setting them to some positive taxation rate with a granularity of $g = 0.2$. This objective and the taxation constraints may be expressed by the formula $\varphi = \mathbf{G}(q = 1 \wedge t_1 \geq 0 \wedge t_2 \geq 0)$.

With static incentive schemes, the only way to achieve this is to set the values of t_1 and t_2 so that at least one of the firms is worse off by performing the d action. If only one firm is taxed in this way, the policy may be criticised as being unfair. If both firms are taxed in this way, this results in an unnecessary loss of profits to both firms. However, if a dynamic incentive scheme is used, which alternates between taxing the firms a sufficient amount for selecting the d action, the regulator can achieve their objective in a fair and efficient manner.

Problem statement. Using the definitions thus far, we are able to state the core problems that we study in the remainder of this text. We begin with the problems in the family of *incentive verification* problems, which ask whether a given static/dynamic incentive scheme guarantees a certain best- or worst-case equilibrium satisfaction value for a given $\mathsf{LTL}[\mathcal{F}]$ formula φ. For a game $\mathcal{G}$, formula φ, static incentive scheme T, solution concept ζ, and threshold $c \in [-1, 1]$.

ζ-S-E-INCENTIVE-VERIFICATION: is there a strategy profile $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$ such that $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

ζ-S-A-INCENTIVE-VERIFICATION: does it hold that for all strategy profiles $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

Similarly, we can define the incentive verification problems for dynamic incentive schemes. For a game $\mathcal{G}$, formula φ, dynamic incentive scheme T, solution concept ζ, threshold $c \in [-1, 1]$.

ζ-D-E-INCENTIVE-VERIFICATION: is there a strategy profile $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$ such that $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

ζ-D-A-INCENTIVE-VERIFICATION: does it hold that for all strategy profiles $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

While the ability to verify the effectiveness of a proposed incentive scheme is arguably a useful tool to have, policymakers are often faced with the task of *designing* effective incentive schemes for a given purpose. This motivates the study of what we refer to as the class of *incentive synthesis* problems, which are defined as follows.

For a given game $\mathcal{G}$, formula φ, solution concept ζ, incentivised variables $U \subseteq$ AP, threshold $c \in [-1, 1]$.

ζ-S-E-INCENTIVE-SYNTHESIS: is there a static incentive scheme T over U such that for some strategy profile $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

ζ-S-A-INCENTIVE-SYNTHESIS: is there a static incentive scheme T over U such that for all strategy profiles $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

ζ-D-E-INCENTIVE-SYNTHESIS: is there a dynamic incentive scheme T over U such that for some strategy profile $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

ζ-D-A-INCENTIVE-SYNTHESIS: is there a dynamic incentive scheme T over U such that for all strategy profiles $\boldsymbol{\sigma} \in \zeta(\mathcal{G})$, we have $[\![\varphi]\!]_{\mathcal{A}[\text{Ag}\mapsto\boldsymbol{\sigma}]}^{G_T} \geq c$?

We remark that although these problems are stated in terms of thresholds, our method of solving these problems allows one to find the optimal satisfaction values of the given formula under some/all equilibria of a game.

4 Static Incentives

In this section, we show how to solve emptiness and synthesis for the static incentive case. We do this by employing procedures for the model checking of formulae in $\mathsf{SL}[\mathcal{F}]$, an extension of $\mathsf{LTL}[\mathcal{F}]$ that includes strategy quantifiers that are useful to quantify over the strategic abilities of agents (Bouyer et al. 2023). We first recall the syntax and semantics of $\mathsf{SL}[\mathcal{F}]$, together with a statement about its model checking against a wCGS. Subsequently, we show how $\mathsf{SL}[\mathcal{F}]$ can be used to express all the various solution concepts considered in this paper. Finally, we show how to use such a representation to solve both the verification and synthesis problems for the case of static incentives.

Definition 5. The syntax of $\mathsf{SL}[\mathcal{F}]$ is defined by the BNF

$$\varphi ::= \psi \mid \exists s.\, \varphi \mid (a, s)\varphi$$

where ψ is an $\mathsf{LTL}[\mathcal{F}]$ formula, $s \in \text{Var}$, and $a \in \text{Ag}$.

Intuitively, the new operator $\exists s.\,\varphi$ selects a strategy that maximizes the satisfaction value of φ; $(a, s)\varphi$ means that the strategy s is assigned to agent a in the evaluation of φ.

Definition 6. Let $G = (\mathrm{AP}, \mathrm{Ag}, (\mathrm{Ac}_a)_{a\in\mathrm{Ag}}, V, \delta, \ell, V_\iota)$ be a wCGS, and $\mathcal{A}$ an assignment. The satisfaction value $[\![\varphi]\!]^G_{\mathcal{A}}(h) \in [-1, 1]$ of an $\mathsf{SL}[\mathcal{F}]$ formula φ in a history h is defined as follows:

$$[\![\exists s.\,\varphi]\!]^G_{\mathcal{A}}(h) = \max_{\sigma\in Str} [\![\varphi]\!]^G_{\mathcal{A}[s\mapsto\sigma]}(h)$$
$$[\![(a, s)\varphi]\!]^G_{\mathcal{A}}(h) = [\![\varphi]\!]^G_{\mathcal{A}[a\mapsto\mathcal{A}(s)]}(h).$$

The other cases are similar to the semantics of $\mathsf{LTL}[\mathcal{F}]$.

We also use the standard univeral quantifiers $\forall s.\,\varphi := \neg\exists s.\,\neg\varphi$. For a variable profile $\boldsymbol{s} = (s_1, ..., s_n)$ with $1 \leq n \leq |\mathrm{Ag}|$, we use the abbreviations $\exists \boldsymbol{s}\,\varphi := \exists s_1, ..., \exists s_n\,\varphi$ and $\forall \boldsymbol{s}\,\varphi := \forall s_1, ..., \forall s_n\,\varphi$.

Definition 7 ($\mathsf{SL}[\mathcal{F}]$ model-checking (Bouyer et al. 2023))**.** Given an $\mathsf{SL}[\mathcal{F}]$ formula φ, a wCGS G, an assignment $\mathcal{A}$, a history h, and a predicate $P \subseteq [-1, 1]$, decide whether $[\![\varphi]\!]^G_{\mathcal{A}}(h) \in P$.

In (Bouyer et al. 2023), it is shown that the model-checking problem for $\mathsf{SL}[\mathcal{F}]$ is decidable and its complexity is $(k+1)$-EXPTIME-complete, where k is the *block-nesting* depth of the checked formula. Informally, the block-nesting depth counts how many times an $\mathsf{SL}[\mathcal{F}]$ formula in prenex-normal form alternates between an existential and a universal quantifier[1].

Expressing solution concepts in $\mathsf{SL}[\mathcal{F}]$. Let $\boldsymbol{s} = (s_a)_{a\in\mathrm{Ag}}$ be a profile of strategy variables and γ_a be a $\mathsf{LTL}[\mathcal{F}]$ formula denoting agent a's goal. The following $\mathsf{SL}[\mathcal{F}]$ formulae express the notion that the strategy profile assigned by $\boldsymbol{\sigma}$ is a strategic equilibrium:

$$\mathrm{DSE}(\boldsymbol{s}) := \bigwedge_{a\in\mathrm{Ag}} \forall \boldsymbol{t}.\left[(\mathrm{Ag}, \boldsymbol{t})\gamma_a \leq (a, s_a)(\mathrm{Ag}_{-a}, \boldsymbol{t}_{-a})\gamma_a\right]$$

$$\mathrm{NE}(\boldsymbol{s}) := \bigwedge_{a\in\mathrm{Ag}} \forall t.\,\left[(\mathrm{Ag}_{-a}, \boldsymbol{s}_{-a})(a, t)\gamma_a \leq (\mathrm{Ag}, \boldsymbol{s})\gamma_a\right]$$

$$\mathrm{RE}_m(\boldsymbol{s}) := \bigwedge_{\mathrm{C}\subseteq\mathrm{Ag}:|\mathrm{C}|\leq m} \forall \boldsymbol{t}_{\mathrm{C}}\Big[\bigwedge_{a\in\mathrm{C}} \big((\mathrm{Ag}_{-\mathrm{C}}, \boldsymbol{s}_{-\mathrm{C}})(\mathrm{C}, \boldsymbol{t}_{\mathrm{C}})\gamma_a \leq (\mathrm{Ag}, \boldsymbol{s})\gamma_a\big)\Big]$$

Proposition 1. *For a wCG $\mathcal{G} = (G, (\gamma_a)_{a\in\mathrm{Ag}})$, a strategy profile $\boldsymbol{\sigma}$ and a variable profile $\boldsymbol{s} = (s_a)_{a\in\mathrm{Ag}}$, it holds that:*

- *$\boldsymbol{\sigma}$ is a dominant strategy equilibrium iff $[\![DSE(\boldsymbol{s})]\!]^G_{\mathcal{A}[a\mapsto\mathcal{A}(s)]}(h) = 1$;*
- *$\boldsymbol{\sigma}$ is a Nash equilibrium iff $[\![NE(\boldsymbol{s})]\!]^G_{\mathcal{A}[a\mapsto\mathcal{A}(s)]}(h) = 1$;*
- *$\boldsymbol{\sigma}$ is an m-resilient equilibrium iff $[\![RE_m(\boldsymbol{s})]\!]^G_{\mathcal{A}[a\mapsto\mathcal{A}(s)]}(h) = 1$.*

[1]By prenex normal form, we mean formulae in which all the quantifiers appear in the outermost part of the formula.

Proof. These follow from the definitions of $\mathrm{util}^G_a(\boldsymbol{\sigma})$ and the solution concepts. □

Using these constructions, we can now address the verification and synthesis problems for static incentives.

Theorem 1. *For $\zeta \in \{DSE, NE, RE_m\}, m \in \{1, ..., n\}$, ζ-S-E-INCENTIVE-VERIFICATION and ζ-S-A-INCENTIVE-VERIFICATION are 2EXPTIME-complete.*

Proof. We begin with ζ-S-E-INCENTIVE-VERIFICATION, noting that the approach for the dynamic counterpart is similar. To establish the upper bound, observe that model checking the following $\mathsf{SL}[\mathcal{F}]$ formula in the game $\mathcal{G}_T$ induced by T decides the ζ-S-E-INCENTIVE-VERIFICATION problem:

$$\exists\boldsymbol{\sigma}.[\zeta(\boldsymbol{\sigma}) \wedge (\mathrm{Ag}, \boldsymbol{\sigma})\varphi].$$

By Proposition 1 and the semantics of the $\mathsf{SL}[\mathcal{F}]$ operators $\exists\boldsymbol{\sigma}$ and $\wedge$, we see that the satisfaction value of the above formula either represents the highest satisfaction value of φ that can be obtained by some ζ-equilibrium of $\mathcal{G}_T$, or it takes on the value of -1. Using this, we can compare the obtained satisfaction value with the threshold c to decide the answer to ζ-S-E-INCENTIVE-VERIFICATION. Membership in 2EXPTIME then straightforwardly follows from the fact that model checking an $\mathsf{SL}[\mathcal{F}]$ formula can be done in $(k + 1) - \mathrm{EXPTIME}$ for formulas φ where the number of alternations in strategy quantifiers is k (Bouyer et al. 2023).

For ζ-S-A-INCENTIVE-VERIFICATION, we can apply the same reasoning as above, only this time with the following $\mathsf{SL}[\mathcal{F}]$ formula:

$$\forall\boldsymbol{\sigma}.[\zeta(\boldsymbol{\sigma}) \rightarrow (\mathrm{Ag}, \boldsymbol{\sigma})\varphi].$$

For hardness, we reduce from qualitative (strong) rational synthesis, which is the problem of deciding whether a strategy profile exists in a concurrent game such that the goal φ_0 of a player 0 is satisfied in some rational outcome of the game while holding this player's strategy fixed (Kupferman, Perelli, and Vardi 2016). Here, the term 'rational' includes the solution concepts we consider, namely dominant strategy equilibria, Nash equilibria, and h-resilient equilibria. The qualitative component means that player goals are given by formulae expressed in LTL, which is a special case of $\mathsf{LTL}[\mathcal{F}]$ (Bouyer et al. 2023), where the following conditions apply:

- For all $v \in V, p \in \mathrm{AP}$, $\ell(v, p) \in \{-1, 1\}$;
- For all players $a \in \mathrm{Ag}$, γ_a is an LTL formula, i.e., an $\mathsf{LTL}[\mathcal{F}]$ formula where $\mathcal{F} = \{\neg, \vee, \wedge\}$.

The strong version of the rational synthesis problem quantifies universally over rational outcomes, i.e., it asks whether there exists a strategy for player 0 such that φ_0 is guaranteed to be satisfied for *all* rational outcomes for the remaining players. Since the (strong) rational synthesis problem is 2EXPTIME-complete for all solution concepts we consider (Kupferman, Perelli, and Vardi 2016), we can straightforwardly obtain the lower bound by reducing from standard rational synthesis to our variants of the E-INCENTIVE-VERIFICATION problem and from strong rational synthesis to the A-INCENTIVE-VERIFICATION problem by simply verifying an incentive scheme which applies no modifications to the game, where the formula to be verified is given by φ_0 and the player whose strategy is fixed is an additional inconsequential dummy player who is added to the game. $\square$

For the synthesis of a static incentive, we can proceed similarly to the emptiness problem. Essentially, we first (nondeterministically) guess an appropriate static incentive, then check that it solves the problem by employing the emptiness procedure introduced above.

Theorem 2. *For $\zeta \in \{DSE, NE, RE_m\}, m \in \{1, ..., n\}$, ζ-S-E-INCENTIVE-SYNTHESIS and ζ-S-A-INCENTIVE-SYNTHESIS are 2EXPTIME-complete.*

Proof. For the upper bound, we proceed by simply nondeterministically guessing an incentive scheme, of which there are $|\Theta| = |\mathrm{AP}|^{1/g}$, where g is the granularity of incentive schemes. Since, g is assumed to be a rational number, the number of incentive schemes is thus exponential in the representation of the granularity.

Static incentive schemes can be represented in space at most equal to that required for representing the valuation function ℓ, which is hence polynomial in the size of the input. Once this incentive scheme is guessed, we can run ζ-S-E-INCENTIVE-VERIFICATION for the existential version of incentive synthesis or ζ-S-A-INCENTIVE-VERIFICATION for the universal version to obtain a solution to our problems. Since guessing a static incentive scheme is in PSPACE and running an incentive verification procedure is in 2EXPTIME for both cases, the overall complexity of the procedure is in 2EXPTIME.

For the lower bound, we can apply the same reduction as used in Proposition 1, but with an additional modification to the game to account for the incentive design component. This can be handled by simply introducing a new propositional variable $q \notin \mathrm{AP}$ to the game, which is guaranteed not to influence the decision-making of agents in any way, and letting the incentive designer intervene only on this new variable, i.e., setting $U = \{q\}$. In this way, there exists an incentive scheme such that the satisfaction value of φ_0 is 1 on some/all ζ-equilibria of the game if and only if the answer to the rational synthesis problem is "yes". $\square$

Given these results, we observe that in the case of static incentives, the incentive verification and synthesis problems have the same worst-case complexity. As our results in the following section will demonstrate, this is not necessarily the case when considering dynamic incentive schemes. This contrast, along with the fact that static incentive schemes are typically simpler to implement and communicate in practice, suggests that it may be better to use static incentives in cases where they are sufficient to implement a goal.

5 Dynamic Incentives

Especially in the context of dynamic incentive schemes, a key aspect of our approach is to embed the incentive designer into the game as a player, whose actions correspond to the application of incentives. To achieve this, we introduce a transformation which converts a given wCGS $\mathcal{G}$ into a modified game $\mathcal{G}'$, which naturally encodes the abilities of the incentive designer to modify the values of propositional variables as described above. Intuitively, the transformation works by interleaving actions of the incentive designer (which correspond to incentive schemes) and the actions of the players in the original game. This is done because the incentive designer requires access to the history of the game in order to decide what incentives to implement on a given round, but this decision must be made *before* the next actions of the agents so that it applies on the round in which they take their actions. To perform this interleaving while preserving the satisfaction values of players' goals on the modified runs, we will first need to introduce the notion of run inflations and formula translations (Kučera and Strejček 2005).

Given an $(\mathrm{AP}, \vec{\ell})$-labelled play $L_1 = L(\pi; \mathrm{AP}, \vec{\ell})$ and an $(\mathrm{AP}', \vec{\ell}')$-labelled play $L_2 = L(\pi'; \mathrm{AP}', \vec{\ell}')$ where $\mathrm{AP} \subseteq \mathrm{AP}'$, we say that L_2 is a *d-fold inflation* of L_1 if it is the case that for all $p \in \mathrm{AP}$, we have $\ell'_{di}(v'_{di}, p) = \ell_i(v_i, p)$ for every $i \in \mathbb{Z}^+$.

We say that a d-fold inflation L_2 of L_1 is *r-labelled* if $r \in \mathrm{AP}'$ and for all $i, j \geq 0$, we have that $\ell'_j(v'_j, r) = 1$ if $j = di$ and $\ell'_j(v'_j, r) = -1$ otherwise. Intuitively, r-labelling is an indicator taking the value of 1 only at the positions in the inflated play π' that correspond to positions in the original play π, which will be useful for translating $\mathsf{LTL}[\mathcal{F}]$ formulae over the latter into formulae over the former.

Next, we define a translation function tr^d which transforms an $\mathsf{LTL}[\mathcal{F}]$ formula φ into another $\mathsf{LTL}[\mathcal{F}]$ formula $\mathsf{tr}^d(\varphi)$ that preserves the satisfaction value of φ over a d-fold inflation of any labelled play over which φ is evaluated. More precisely, let φ be an $\mathsf{LTL}[\mathcal{F}]$ formula over a set AP of atomic propositions. The *d-fold translation* $\mathsf{tr}^d(\varphi)$ of φ is the $\mathsf{LTL}[\mathcal{F}]$ formula defined over the set $\mathrm{AP} \cup \{r\}$, where r is a new atomic proposition, that is given by the following semantics:

- $\mathsf{tr}^d(p) = \mathbf{X}^{d-1}p$ for every $p \in \mathrm{AP}$;
- $\mathsf{tr}^d(f[\varphi_1, ..., \varphi_m]) = f[\mathsf{tr}^d(\varphi_1), ..., \mathsf{tr}^d(\varphi_m)]$;
- $\mathsf{tr}^d(\mathbf{X}\varphi) = \mathbf{X}^d \mathsf{tr}^d(\varphi)$;
- $\mathsf{tr}^d(\varphi_1 \mathbf{U} \varphi_2) = (r \to \mathsf{tr}^d(\varphi_1))\mathbf{U}(r \to \mathsf{tr}^d(\varphi_2))$

where X^e represents an application of the $\mathbf{X}$ operator e times. The following is an adaptation of Lemma 1 in (Gutierrez, Perelli, and Wooldridge 2018) to the setting of $\mathsf{LTL}[\mathcal{F}]$ formulae that we consider.

Lemma 1. *Let AP and AP$''$ be two disjoint sets of atomic propositions with $r \in \mathrm{AP}''$. Let $L_1 = L(\pi; \mathrm{AP}, \vec{\ell})$ be an $(\mathrm{AP}, \vec{\ell})$-labelled play and suppose that $L_2 = L(\pi'; \mathrm{AP}', \vec{\ell}')$ is an r-labelled, d-fold inflation of L_1 with $\mathrm{AP}' = \mathrm{AP} \cup \mathrm{AP}''$. Then, for all $\mathsf{LTL}[\mathcal{F}]$ formulae φ over L_1, it holds that $[\![\varphi]\!]_{\pi}^{\mathrm{AP}, \vec{\ell}} = [\![\mathsf{tr}^d(\varphi)]\!]_{\pi'}^{\mathrm{AP}', \vec{\ell}'}$.*

Proof. The proof goes by structural induction on the formula φ, that is, we show that for all $\mathsf{LTL}[\mathcal{F}]$ formula φ and

$i \geq 0$, it holds that $[\![\varphi]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}} = [\![\mathsf{tr}^d(\varphi)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}}$. Considering the base case, it is straightforward to see that $[\![p]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq 1}} = [\![\mathsf{tr}^d(p)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d-1}}$ by the definition of the d-fold inflation. Now, by the induction hypothesis and the definition of the semantics for $[\![f[\varphi_1,..., \varphi_m]]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}}$, it follows that

$$\begin{aligned}[\![\mathsf{tr}^d(f[\varphi_1,..., \varphi_m])]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}} &= [\![f[\mathsf{tr}^d(\varphi_1),..., \mathsf{tr}^d(\varphi_m)]]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}} \\ &= f([\![\mathsf{tr}^d(\varphi_1)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}},..., [\![\mathsf{tr}^d(\varphi_m)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}}) \\ &= f([\![\varphi_1]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}},..., [\![\varphi_m]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}}) = [\![f[\varphi_1,..., \varphi_m]]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}}.\end{aligned}$$

Moving on to the next operator, suppose that $\varphi = \mathbf{X}\psi$ for some $\mathsf{LTL}[\mathcal{F}]$ formula ψ and recall that by definition, $[\![\mathbf{X}\psi]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}} = [\![\psi]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i+1}}$. By the semantics of the $\mathbf{X}$ operator and the induction hypothesis, we have

$$\begin{aligned}[\![\mathbf{X}\psi]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}} = [\![\psi]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i+1}} &= [\![\mathsf{tr}^d(\psi)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d(i+1)}} \\ &= [\![\mathbf{X}^d\mathsf{tr}^d(\psi)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}} = [\![\mathsf{tr}^d(\mathbf{X}\psi)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}}.\end{aligned}$$

Finally, for the until operator, we apply the same reasoning. Suppose that $\varphi = \varphi_1\mathbf{U}\varphi_2$. Then, by the semantics of the $\mathbf{U}$ operator and the induction hypothesis, we have

$$\begin{aligned}[\![\varphi_1\mathbf{U}\varphi_2]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i}} &= \sup_{j\geq 0}\min\big([\![\varphi_2]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i+j}}, \min_{0\leq k<j}[\![\varphi_1]\!]^{\mathrm{AP},\vec{\ell}}_{\pi_{\geq i+k}}\big) \\ &= \sup_{j\geq 0}\min\big([\![\mathsf{tr}^d(\varphi_2)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d(i+j)}}, \\ &\qquad \min_{0\leq k<j}[\![\mathsf{tr}^d(\varphi_1)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d(i+k)}}\big) \\ &= \sup_{j\geq 0}\min\big([\![\max(-r, \mathsf{tr}^d(\varphi_2))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d(i+j)}}, \\ &\qquad \min_{0\leq k<j}[\![\max(-r, \mathsf{tr}^d(\varphi_1))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq d(i+k)}}\big) \\ &= \sup_{j\geq 0}\min\big([\![\max(-r, \mathsf{tr}^d(\varphi_2))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di+j}}, \\ &\qquad \min_{0\leq k<j}[\![\max(-r, \mathsf{tr}^d(\varphi_1))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di+k}}\big) \\ &= \sup_{j\geq 0}\min\big([\![r \to \mathsf{tr}^d(\varphi_2))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di+j}}, \\ &\qquad \min_{0\leq k<j}[\![r \to \mathsf{tr}^d(\varphi_1))]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di+k}}\big) \\ &= [\![\mathsf{tr}^d(\varphi_1\mathbf{U}\varphi_2)]\!]^{\mathrm{AP}',\vec{\ell}'}_{\pi'_{\geq di}}.\end{aligned}$$

□

Together, the concepts of inflation and translation allow us to define the *incentive-augmented game* $\mathcal{G}'$, which is a modified version of a game $\mathcal{G}$ which includes the incentive designer as a player whose actions represent the imposition of different incentive schemes. Given a wCG $\mathcal{G} = (\mathrm{AP}, \mathrm{Ag}, (\mathrm{Ac}_a)_{a\in\mathrm{Ag}}, V, v_\iota, \delta, \ell, (\gamma_a)_{a\in\mathrm{Ag}})$, the *incentive-augmented* version of $\mathcal{G}$ is defined as the wCG $\mathcal{G}' = (\mathrm{AP}', \mathrm{Ag}', (\mathrm{Ac}_a)_{a\in\mathrm{Ag}'}, V', v_\iota, \delta', \ell', (\gamma'_a)_{a\in\mathrm{Ag}})$, where

- $\mathrm{AP}' = \mathrm{AP} \cup \{r\}$, where $r \notin \mathrm{AP}$.
- $\mathrm{Ag}' = \mathrm{Ag} \cup \{n+1\}$, where player $n+1$ corresponds to the incentive designer;
- Ac_a remains unchanged for all $a \in \mathrm{Ag}$, and $\mathrm{Ac}_{n+1} = \Theta$;
- $V' = V \cup V^\Theta$, where $V^\Theta = \{v^\theta | v \in V, \theta \in \Theta\}$;
- $\delta' : V' \times \bigtimes_{a\in\mathrm{Ag}'}\mathrm{Ac}'_a \to V'$ is defined in the following way. For all $v \in V, \boldsymbol{c}' \in \mathrm{Ac}'$ with $\boldsymbol{c}'_{n+1} = \theta \in \Theta$, we have $\delta'(v, \boldsymbol{c}') = v^\theta$ and for all $v \in V^\Theta, \boldsymbol{c}' \in \mathrm{Ac}'$ with $\boldsymbol{c}'_{-n+1} = (c_a)_{a\in\mathrm{Ag}}$, we have $\delta'(v^\theta, \boldsymbol{c}') = \delta(v, (c_a)_{a\in\mathrm{Ag}})$;
- $\ell' : V' \times \mathrm{AP} \to [-1, 1]$ is defined such that for all $v \in V, p \in \mathrm{AP}$, we have $\ell'(v, p) = 0$ and for all $v^\theta \in V^\Theta$, we have
$$\ell'(v^\theta, p) = \begin{cases}\theta(p), & p \in U \\ \ell(v, p), & p \notin U.\end{cases}$$
Moreover, for all $v \in V'$, we have
$$\ell'(v, r) = \begin{cases}-1, & v \in V \\ 1, & v \in V^\Theta;\end{cases}$$
- $\gamma'_a = \mathsf{tr}^2(\gamma_a)$.

This translation of goals is well-defined due to the addition of the r variable which is used to r-label plays in $\mathcal{G}'$. Intuitively, $\mathcal{G}'$ introduces a new player – the 'incentive player' – whose actions correspond to the application of incentives at each round of the original game. The incentive player's actions are taken in an alternating manner with the rest of the players in the positions V, which are interleaved with the new positions V^Θ such that at every odd timestep $t = 2k+1, k \in \mathbb{N}$, the game will be in a position $v_t \in V$, whereas at every positive even timestep $t' = 2k, k \in \mathbb{Z}^+$, the game will be in a position $v^\theta_{t'} \in V^\Theta$. In this way, at positions $v \in V$, only the incentive player's action determines the next position and at positions $v^\Theta \in V^\Theta$, only the original players' actions (i.e., players $1, ..., n$) will determine the next position.

Due to the definition of the modified valuation function ℓ' and the 2-fold translation of the players' objectives, the incentive schemes chosen by the incentive player at each of their turns modify the satisfaction values of the players' objectives, influencing the players' strategic considerations.

Moreover, it is straightforward to see that once the incentive player's strategy has been fixed, this induces a new game with potentially different stable outcomes from the original game. Under this construction, the task of the incentive player can then be formulated as the problem of choosing a strategy such that some or all of the induced equilibria of the resulting game for the remaining players maximises the satisfaction value of the incentive designer's objective. The following result establishes the relationship between strategies for the incentive player in an incentive-augmented game and the application of dynamic incentive schemes in the original game.

Proposition 2. *Suppose that $\mathcal{G}$ is a weighted concurrent game, $\mathcal{G}'$ is its incentive-augmented version, φ is an $\mathsf{LTL}[\mathcal{F}]$ formula, and $(\boldsymbol{\sigma}', \mathcal{T})$ is a strategy profile in $\mathcal{G}'$. Then, there* *is a strategy profile $\boldsymbol{\sigma}$ in $\mathcal{G}$ and a dynamic incentive scheme T such that*

$$[\![\varphi]\!]^{G_T}_{\mathcal{A}[\mathrm{Ag}\mapsto(\boldsymbol{\sigma})]} = [\![\mathsf{tr}^2(\varphi)]\!]^{G'}_{\mathcal{A}[\mathrm{Ag}'\mapsto(\boldsymbol{\sigma}',\mathcal{T})]}.$$

Proof. We prove the claim by constructing the strategy profile $\boldsymbol{\sigma}$ and the dynamic incentive scheme T. Firstly, let $\boldsymbol{\sigma}$ be the strategy profile in $\mathcal{G}$ such that for all even-length histories $h' = v_1 v_1^{\theta_1} v_2 v_2^{\theta_2} ... v_k v_k^{\theta_k}, k \in \mathbb{Z}^+$ that are generated by $\boldsymbol{\sigma}'$ in $\mathcal{G}'$, it holds that $\boldsymbol{\sigma}(h) = \boldsymbol{\sigma}'(h')$, where $h = v_1 v_2, ..., v_k$. This strategy profile will be referred to as the *2-fold deflation* of the strategy profile $\boldsymbol{\sigma}'$, which extracts the actions that players take at every second round of the game $\mathcal{G}'$. Next, let T be the dynamic incentive scheme such that for any given history $h' = v_1 v_1^{\theta_1} ... v_{k-1}^{\theta_{k-1}} v_k$ of a play π' in $\mathcal{G}'$ and the corresponding history $h = v_1 v_2 ... v_k$ of a play π in $\mathcal{G}$ resulting from the execution of $\boldsymbol{\sigma}$, we have that $\mathcal{T}(h') = T(h)$. By the construction of the game $\mathcal{G}'$, we see that the incentive scheme θ_k corresponding to the chosen action $\mathcal{T}(h')$ is implemented in position $v_k^{\theta_k}$, that is, for all $p \in \mathrm{AP}$, we have $\ell'(v_k^{\theta_k}, p) = \vec{\ell}_T(h) = \ell_{\theta^k}$. In other words, the satisfaction values for all variables on round $2k, k \in \mathbb{Z}^+$ of the game $\mathcal{G}'$ coincide with those on round k of the game $\mathcal{G}$.

Now, let π' be the play generated in $\mathcal{G}'$ by $(\boldsymbol{\sigma}', \mathcal{T})$, $\vec{\ell'} = \ell'\ell'...$, π be the play generated in $\mathcal{G}$ by $(\boldsymbol{\sigma})$, and $\vec{\ell}_T(\pi) = \ell_{\theta_1}\ell_{\theta_2}...$. Then, we can conclude that the $(\mathrm{AP}', \vec{\ell'})$-labelled play $L_2 = L(\pi'; \mathrm{AP}', \vec{\ell'})$ in $\mathcal{G}'$ is an r-labelled d-fold inflation of $L_1 = L(\pi; \mathrm{AP}, \vec{\ell}_T(\pi))$ in $\mathcal{G}$ and hence, the result follows by the application of Lemma 1. □

Finally, we turn our attention to the dynamic incentive verification and synthesis problems. Again, we are able to make use of the expressive power of $\mathsf{SL}[\mathcal{F}]$ to define a suitable formula which can be model-checked to determine the solution to our problems.

Theorem 3. *For $\zeta \in \{DSE, NE, RE_m\}, m \in \{1, ..., n\}$, ζ-*D-E-INCENTIVE-VERIFICATION *and ζ-*D-A-INCENTIVE-VERIFICATION *are 2EXPTIME-complete.*

Proof. Regarding ζ-D-E-INCENTIVE-VERIFICATION, we use the construction $\mathcal{G}'$, which introduces the incentive designer as a player in the game, whose actions are interleaved with those of the original players Ag. Given this, we can solve the incentive verification problem by model checking the following $\mathsf{SL}[\mathcal{F}]$ formula in the game $\mathcal{G}'$:

$$(n+1, \mathcal{T})\ \big[\exists \boldsymbol{\sigma}'.[\zeta(\boldsymbol{\sigma}') \wedge (\mathrm{Ag}, \boldsymbol{\sigma}')\mathsf{tr}^2(\varphi)]\big],$$

where the strategy $\mathcal{T}$ is defined such that for all histories $h = v_1 v_2 ... v_k \in$ Hist of the original game $\mathcal{G}$ and a corresponding history $h' = v_1 v_1^{\theta_1} v_2 v_2^{\theta_2} ... v_k^{\theta_k}$, we have that $\mathcal{T}(h') = T(h)$. By the construction of $\mathcal{G}'$, strategies for the original players in Ag are inconsequential on odd timesteps, and moreover, the formula $\zeta(\boldsymbol{\sigma}')$ is now defined in terms of the 2-fold translated versions of players' goals. Applying the definition of the 2-fold translation of

Thus, by Proposition 2, $\zeta(\boldsymbol{\sigma}')$ has a satisfaction value of 1 in the above formula if and only if the 2-fold deflation of $\boldsymbol{\sigma}'$ in the original game $\mathcal{G}_T$ is a ζ-equilibrium. Moreover, by Proposition 2 again, the satisfaction value of the subformula $(\mathrm{Ag}, \boldsymbol{\sigma}')\mathsf{tr}^2(\varphi)$ in $\mathcal{G}'$ is equal to the satisfaction value of the formula $(\mathrm{Ag}, \boldsymbol{\sigma})\varphi$ in the game $\mathcal{G}_T$. Hence, model-checking this formula indeed solves the ζ-D-E-INCENTIVE-VERIFICATION problem. For all of our solution concepts, this formula has an alternation depth of 1, and hence, the formula can be model-checked in 2EXPTIME.

Similarly, for ζ-D-A-INCENTIVE-VERIFICATION, we model check the formula:

$$(n+1, \mathcal{T})\ \big[\forall \boldsymbol{\sigma}'.[\zeta(\boldsymbol{\sigma}') \rightarrow (\mathrm{Ag}, \boldsymbol{\sigma}')\mathsf{tr}^2(\varphi)]\big],$$

which again has an alternation depth of 1 when converted into prenex normal form.

For hardness, we can apply the same reduction as used in the proof of Proposition 1, i.e., we reduce from qualitative rational synthesis and its strong variant to the ζ-D-E-INCENTIVE-VERIFICATION and ζ-D-A-INCENTIVE-VERIFICATION problems respectively. Again, we check a dynamic incentive scheme that does not modify any of the variables and use player 0's goal as the global formula to be checked, with a threshold $c = 1$. □

As with static incentive schemes, the incentive verification problem for dynamic incentive schemes remains in 2EXPTIME. However, the following result indicates that we do not necessarily obtain the same complexity when moving from static to dynamic incentive synthesis.

Theorem 4. *For $\zeta \in \{DSE, NE, RE_m\}, m \in \{1, ..., n\}$, ζ-*D-E-INCENTIVE-SYNTHESIS *is 2EXPTIME-complete and ζ-*D-A-INCENTIVE-SYNTHESIS *is in 3EXPTIME and is 2EXPTIME-hard.*

Proof. Starting with the existential version of the problem, we again make use of the construction $\mathcal{G}'$. The ζ-D-E-INCENTIVE-SYNTHESIS problem can be solved by model-checking the following $\mathsf{SL}[\mathcal{F}]$ formula:

$$\exists \mathcal{T}.\ \big[(n+1, \mathcal{T})\exists \boldsymbol{\sigma}'.[\zeta(\boldsymbol{\sigma}') \wedge (\mathrm{Ag}, \boldsymbol{\sigma}')\mathsf{tr}^2(\varphi)]\big].$$

By a similar argument as in the proof of Theorem 3, model-checking this formula solves the ζ-D-E-INCENTIVE-SYNTHESIS, and we can extract the dynamic incentive scheme T from a witness $\mathcal{T}$ to the outer existential quantifier of the above formula using the method outlined in the proof of Proposition 2. Noting that the quantifiers over $\mathcal{T}$ and $\boldsymbol{\sigma}'$ do not introduce another level of alternation, model checking this formula is also in 2EXPTIME.

For ζ-D-A-INCENTIVE-SYNTHESIS, we cannot group the quantifiers for incentive schemes and the candidate ζ-equilibria, which introduces another level of alternation in the representative $\mathsf{SL}[\mathcal{F}]$ formula:

$$\exists \mathcal{T}.\ \big[(n+1, \mathcal{T})\forall \boldsymbol{\sigma}'.[\zeta(\boldsymbol{\sigma}') \rightarrow (\mathrm{Ag}, \boldsymbol{\sigma}')\mathsf{tr}^2(\varphi)]\big].$$

This leads to a 3EXPTIME procedure for the universal incentive synthesis problem. For the lower bounds, one can use the same reduction as in Theorem 2, where we set up the game so that the incentive designer has no effect on the rational outcomes of the original rational synthesis problem. The addition of memory for the incentive designer does not impact their capabilities in this setup, so the arguments for the reduction carry through. □

6 Conclusion

We have introduced an approach to designing incentives for rational agents with $\mathsf{LTL}[\mathcal{F}]$ goals in order to achieve a societal objective. We formalized the problems of verifying and synthesizing static and dynamic incentive schemes for a range of solution concepts. We then proved that, for all solution concepts that we consider, the complexity of incentive verification is 2EXPTIME-complete, which is not harder than the corresponding qualitative rational verification problems. In practice, verification is double exponential only in the sizes of the formulae representing the societal objectives, which are typically small for most applications. In the case of incentive synthesis, we showed it is 2EXPTIME-complete in the static case and in 3EXPTIME in the dynamic case.

Similar to the rational verification problem, our static incentive synthesis has double exponential cost only in the size of the goals. However, there is also an additional exponential cost in the size of the representation of the granularity, which can be chosen by the incentive designer for their purposes. In practical applications such as setting interest rates or fixing minimum bidding increments in auctions, incentive designers typically use a rather coarse level of granularity, which would not incur a significant blowup in the algorithm's worst-case runtime. For dynamic incentive synthesis, there is a triple exponential cost in the size of the goals, and we again have an exponential cost in the size of the granularity, but the cost is the same as in verification for the remaining parts of the input. Thus, even with a potentially exponential gap between static and dynamic incentive synthesis, for relatively succinct goal specifications, there may not be a significant difference in runtimes between the two approaches.

For future work, we intend to study the synthesis of incentive schemes with minimal modifications to the original game and to analyze Incentive Design in the cooperative setting. Due to the expressivity of $\mathsf{SL}[\mathcal{F}]$, many additional requirements or constraints that one may wish to impose on the incentive schemes can be naturally incorporated into the global goal in our framework, which enables a wide variety of incentive design problems to be solved using the approaches outlined here.

Acknowledgements

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101105549, the PNRR MUR project PE0000013-FAIR, and the PRIN 2020 projects PINPOINT and RIPER (No. 20203FFYLK). Giuseppe Perelli was also supported by Sapienza University of Rome under the "*Progetti Grandi di Ateneo*" programme, grant RG123188B3F7414A (ASGARD - Autonomous and Self-Governing Agent-Based Rule Design). Michael Wooldridge was supported by the UKRI under a Turing AI World Leading Research Fellowship (EP/W002949/1).

References

Abraham, I.; Dolev, D.; and Halpern, J. Y. 2008. Lower bounds on implementing robust and resilient mediators. In *Theory of Cryptography: Fifth Theory of Cryptography Conference*, 302–319. Springer.

Alechina, N.; De Giacomo, G.; Logan, B.; and Perelli, G. 2022. Automatic synthesis of dynamic norms for multi-agent systems. In *Proc. of the Int. Conference on Principles of Knowledge Representation and Reasoning*, volume 19, 12–21.

Almagor, S.; Avni, G.; and Kupferman, O. 2015. Repairing multi-player games. In *Proc. of the Int. Conference on Concurrency Theory, CONCUR 2015*, volume 42 of *LIPIcs*, 325–339. Schloss Dagstuhl - Leibniz-Zentrum für Informatik.

Almagor, S.; Boker, U.; and Kupferman, O. 2016. Formally reasoning about quality. *Journal of the ACM* 63(3):24:1–24:56.

Almagor, S.; Kupferman, O.; and Perelli, G. 2018. Synthesis of controllable nash equilibria in quantitative objective game. In *Proc. of the Int. Joint Conference on Artificial Intelligence, IJCAI 2018*, 35–41. ijcai.org.

Alur, R.; Henzinger, T.; and Kupferman, O. 2002. Alternating-time temporal logic. *Journal of the ACM* 49(5):672–713.

Baier, C., and Katoen, J. 2008. *Principles of model checking*. MIT Press.

Belardinelli, F.; Jamroga, W.; Malvone, V.; Mittelmann, M.; Murano, A.; and Perrussel, L. 2022. Reasoning about human-friendly strategies in repeated keyword auctions. In *Proc. of the Int. Conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2022*, 62–71. IFAAMAS.

Bouyer, P.; Kupferman, O.; Markey, N.; Maubert, B.; Murano, A.; and Perelli, G. 2023. Reasoning about Quality and Fuzziness of Strategic Behaviors. *ACM Trans. Comput. Log.* 24(3):21:1–21:38.

Brice, L.; Raskin, J.; and van den Bogaard, M. 2023. Rational Verification for Nash and Subgame-Perfect Equilibria in Graph Games. In Leroux, J.; Lombardy, S.; and Peleg, D., eds., *Proc. of the Int. Symposium on Mathematical Foundations of Computer Science, MFCS 2023*.

Centeno, R., and Billhardt, H. 2011. Using incentive mechanisms for an adaptive regulation of open multi-agent systems. In *Proc. of the Int. Joint Conference on Artificial Intelligence, IJCAI 2011*.

Condurache, R.; Filiot, E.; Gentilini, R.; and Raskin, J. 2016. The Complexity of Rational Synthesis. In *Int. Colloquium on Automata, Languages, and Programming, ICALP 2016*.

Filiot, E.; Gentilini, R.; and Raskin, J. 2018. Rational Synthesis Under Imperfect Information. In Dawar, A., and Grädel, E., eds., *Proc. of the Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2018*.

Fisman, D.; Kupferman, O.; and Lustig, Y. 2010. Rational synthesis. In Esparza, J., and Majumdar, R., eds., *Tools and Algorithms for the Construction and Analysis of Systems*.

Gutierrez, J.; Perelli, G.; and Wooldridge, M. J. 2018. Imperfect information in reactive modules games. *Information and Computation* 261:650–675.

Hyland, D.; Gutierrez, J.; and Wooldridge, M. 2023a. Principal-agent boolean games. In *Proc. of the Int. Joint Conference on Artificial Intelligence, IJCAI 2023*, 144–152. ijcai.org.

Hyland, D.; Gutierrez, J.; and Wooldridge, M. J. 2023b. Incentive engineering for concurrent games. In *Proc. of the Conference on Theoretical Aspects of Rationality and Knowledge, TARK 2023*, volume 379 of *EPTCS*, 344–358.

Kupferman, O., and Shenwald, N. 2022. The Complexity of LTL Rational Synthesis. In Fisman, D., and Rosu, G., eds., *Int. Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2022*. Springer.

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

Kučera, A., and Strejček, J. 2005. The stuttering principle revisited. *Acta Informatica* 41(7–8):415–434.

Laffont, J.-J., and Martimort, D. 2009. The theory of incentives: the principal-agent model. In *The theory of incentives*. Princeton University Press.

Maubert, B.; Mittelmann, M.; Murano, A.; and Perrussel, L. 2021. Strategic reasoning in automated mechanism design. In *Proc. of the Int. Conference on Knowledge Representation and Reasoning, KR 2021*.

Mittelmann, M.; Maubert, B.; Murano, A.; and Perrussel, L. 2022. Automated synthesis of mechanisms. In *Proc. of the International Joint Conference in Artificial Intelligence, IJCAI 2022*, 426–432. ijcai.org.

Narayanasamy, S. K. 2022. Automating mechanism design with program synthesis. In *Proc. Automated Learning Agents Workshop*.

Nisan, N.; Roughgarden, T.; Tardos, É.; and Vazirani, V. 2007. *Algorithmic Game Theory*. Cambridge University Press.

Olsder, G. 2009. Phenomena in inverse stackelberg games, part 1: Static problems. *Journal of optimization theory and applications* 143:589–600.

Parkes, D. C.; Cavallo, R.; Constantin, F.; and Singh, S. 2010. Dynamic incentive mechanisms. *AI Magazine* 31(4):79–94.

Pnueli, A. 1977. The temporal logic of programs. In *Proc. of the Annual Symposium on Foundations of Computer Science, SFCS 1977*, 46–57. IEEE.

Ratliff, L. J., and Fiez, T. 2020. Adaptive incentive design. *IEEE Transactions on Automatic Control* 66(8):3871–3878.

Ratliff, L. J.; Dong, R.; Sekar, S.; and Fiez, T. 2019. A perspective on incentive design: Challenges and opportunities. *Annual Review of Control, Robotics, and Autonomous Systems* 2:305–338.

Weber, T. A. 2011. *Optimal control theory with applications in economics*. MIT press.

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

Yang, J.; Wang, E.; Trivedi, R.; Zhao, T.; and Zha, H. 2022. Adaptive incentive design with multi-agent meta-gradient reinforcement learning. In *Proc. of the Int. Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022*.