Non-Obvious Manipulability in Extensive-Form Mechanisms: The Revelation Principle for Single-Parameter Agents

Thomas Archbold , **Bart de Keijzer** and **Carmine Ventre**
King's College London
{thomas.archbold,bart.de_keijzer,carmine.ventre}@kcl.ac.uk

Abstract

Recent work in algorithmic mechanism design focuses on designing mechanisms for agents with *bounded rationality*, modifying the constraints required to achieve incentive compatibility. Starting with Li's strengthening of strategyproofness, *obvious strategyproofness (OSP)* requires truthtelling to be "obvious" over dishonesty, roughly meaning that the worst outcome from truthful actions must be no worse than the best outcome for dishonest ones. A celebrated result for dominant-strategy incentive-compatible mechanisms that allows us to restrict attention to direct mechanisms, known as the *revelation principle*, does not hold for OSP: the implementation details matter for the obvious incentive properties of the mechanism. Studying agent strategies in real-life mechanisms, Troyan and Morrill introduce a relaxation of strategyproofness known as non-obvious manipulability, which only requires comparing certain extrema of the agents' utility functions in order for a mechanism to be incentive-compatible: a mechanism is *not obviously manipulable (NOM)* if the best and worst outcomes when acting truthfully are no worse than the best and worst outcomes when acting dishonestly. In this work we first extend the cycle monotonicity framework for direct-revelation NOM mechanism design to indirect mechanisms. We then apply this to two settings, single-parameter agents and mechanisms for two agents in which one has a two-type domain, and show that here the revelation principle holds: direct mechanisms are just as powerful as indirect ones.

1 Introduction

Traditional algorithm design focuses on mapping inputs to outputs under some notion of efficiency. In addition to this algorithmic mechanism design assumes that the inputs must first be gathered from a set of agents who each possess some private information (or "type") about the true state of the world. Since these agents can influence the output of the algorithm depending on how they report their type it may be beneficial to lie. As the mechanism designer we must realign the incentives of the agents to ensure they are acting in the desired manner, with respect to some solution concept. A mechanism that guarantees this behaviour is said to be incentive-compatible. We study mechanisms that use monetary transfers to achieve incentive-compatibility.

Strategyproofness is a solution concept that stipulates it is a dominant strategy for each agent to report their type truthfully to the mechanism, for each joint action of the other players. This is appealing from a theoretical standpoint since a strategyproof mechanism eliminates the possibility for any agent to benefit from a unilateral deviation. This relies on the assumption that agents are perfectly rational and therefore correctly reason about all possible actions to conclude that truthtelling is dominant. Such reasoning may place a high cognitive burden on the agent, so it may be unrealistic to expect agents to behave "correctly" in strategyproof mechanisms in practice. Empirical evidence ([Roth, 1991]) suggests that agents may fail to recognise the benefits of misreporting in mechanisms based on the Deferred Acceptance algorithm, even though they are manipulable, while others such as the Boston Mechanism ([Dur *et al.*, 2018]) are more readily manipulated. [Troyan and Morrill, 2020] compare these mechanisms and argue that deciding when to manipulate in the former is much easier than in the latter: in the Boston Mechanism an agent can always guarantee a place her second-choice school, but in the Deferred Acceptance algorithm determining a profitable manipulation requires a detailed understanding of the mechanism and the other agents' preferences. Focusing on direct mechanisms, they introduce a model of bounded rationality where agents can only recognise manipulations that are "obvious", where an agent's best or worst outcome when misreporting her type to the mechanism is strictly greater than her best or worst outcome when reporting her type truthfully, and the best and worst cases are taken over all joint actions of the other agents. If no agent has an obvious manipulation then the mechanism is said to be *not obviously manipulable (NOM)*. While a NOM mechanism may be manipulable, we are safe to ignore the non-obvious manipulations since they will not be recognised by these cognitively-limited agents.

The mechanisms we study use monetary transfers. Any mechanism must first collect a type profile from the agents before returning an outcome and a vector of payments, but there is flexibility in how the mechanism determines the type profile. In a direct-revelation mechanism each agent reports her type directly to the mechanism, for example by writing down a monetary value for the item on offer in a single-item sealed-bid auction. In general agents need not report their types directly; instead, the mechanism can deduce the type profile by querying the agents over a number of rounds. In the ascending-clock auction, for example, bidders gradually drop out of contention for the item on sale as the sale price increases, with the item being awarded to the sole remaining bidder at the final price displayed. In addition to the outcome and payment vectors it returns, an indirect or extensive-form mechanism is defined by an implementation tree which specifies how to elicit the type profile from the agents.

We study which social choice functions are *implementable* by indirect mechanisms with respect to the NOM constraints: given some social choice function that maps type profiles to outcomes, can we define a payment rule and an implementation tree such that the resulting mechanism is incentive-compatible? In an indirect mechanism agents are not limited to acting only once, and throughout the mechanism's execution they may infer information about the the other agents' types which they would otherwise be unable to (partially) observe under a direct mechanism. It is therefore natural to ask whether indirect mechanisms are more powerful than direct ones in achieving incentive-compatibility, or: do the implementation details affect which social choice functions are implementable? For strategyproofness the revelation principle says that every indirect mechanism can be simulated by a direct mechanism (see, e.g., [Nisan *et al.*, 2007, ch. 9]). In this work we show that the revelation principle holds for two settings under NOM, partially answering an open question raised by [Troyan and Morrill, 2020]. These can be applied naturally to a range of contexts, and our results can be interpreted in two ways: on the one hand restricting attention to direct mechanisms means there is a smaller design space to consider when constructing NOM incentive-compatible mechanisms; on the other hand, if a social choice function is not implementable by a direct mechanism, then nor will it be by an indirect one.

1.1 Related Work

There has been growing interest in the study of mechanism design for agents with bounded rationality. [Li, 2017] strengthens the notion of strategyproofness by introducing OSP as a solution concept for agents that lack contingent reasoning skills, arguing agents will only tell the truth if it is obvious over lying. OSP has since been applied to a range of settings including combinatorial auctions [de Keijzer *et al.*, 2020], machine scheduling [Ferraioli *et al.*, 2019], and facility location [Ferraioli and Ventre, 2021]. Importantly the revelation principle does not hold for OSP and the implementation details are vital to whether the mechanism is incentive-compatible.

[Troyan and Morrill, 2020] take a more optimistic perspective and relax strategyproofness to non-obvious manipulability, motivated by empirical observations that manipulations are easier to recognise in certain mechanisms in practice than others. After formalising what it means for a manipulation to be obvious they characterise NOM mechanisms in the context of school choice, two-sided matching, auctions, and bilateral trade, and use this to classify several existing mechanisms as either obviously manipulable or not.

NOM has been applied to a range of other settings. [Aziz and Lam, 2021] consider NOM voting rules where agents are equipped with ordinal (instead of cardinal) preferences and partially characterise certain classes of NOM voting rules and show that the number of agents may affect the obvious incentive properties of a mechanism. They also consider the complexity of computing obvious manipulations in certain voting rules by providing reductions to the Constructive Coalitional Unweighted Matching problem, yielding a polynomial-time algorithm for the k approval voting rule.

[Ortega and Segal-Halevi, 2022] study NOM in conjunction with a particular class of extensive-form mechanisms without money for cake-cutting known as Robertson-Webb mechanisms. A drawback of direct mechanisms in this setting is that they only make sense when valuations can be succinctly represented, hence the focus on indirect mechanisms. They show that the conflict between incentive-compatibility and proportionality is resolved when relaxing strategyproofness to NOM, and that every proportional direct mechanism is NOM for worst cases while every Pareto Optimal direct mechanism is NOM for best cases.

[Psomas and Verma, 2022] study NOM mechanisms without money for fairly allocating divisible goods among agents with additive preferences and show that there exist deterministic direct mechanisms achieving envy-freeness up to one good (EF1). They show that optimising against different objective functions may affect a mechanism's obvious manipulability: while there is a NOM mechanism maximising social welfare for $n \geq 3$ agents, it is impossible for a NOM mechanism to achieve optimal egalitarian or Nash social welfare. They also reduce the problem of designing NOM and EF1 mechanisms to that of designing EF1 algorithms while preserving efficiency.

[Archbold *et al.*, 2023] study the design of direct-revelation NOM mechanisms with monetary transfers. They characterise the set of implementable social choice functions in this setting using cycle-monotonicity then restrict attention to characterise the implementable social choice functions for single-parameter agents, along with their payments. Studying an impossibility result from [Troyan and Morrill, 2020] they show that any efficient and individually rational mechanism for bilateral trade requires an unbounded subsidy for a class of simple "single-line" mechanisms, and show that this is inevitable when considering only best cases but can be avoided when only considering worst cases. Additional NOM mechanisms have been studied in the context of matching [Troyan *et al.*, 2020], assignment [Troyan, 2022], and energy markets [Kiedanski *et al.*, 2020].

[Bade and Gonczarowski, 2017] study an analogue of the revelation principle for mechanisms for boundedly-rational agents, providing a "gradual revelation principle" for OSP mechanisms which states that every extensive-form mechanism can be transformed into a certain extensive-form mechanism where agents gradually reveal more about their preferences.

1.2 Our Contribution

We begin by investigating the social choice functions that can be implemented by indirect NOM mechanisms by extending the concept of labellings for direct mechanisms from [Archbold *et al.*, 2023] to handle extensive-form implementations. We use the well-known cycle-monotonicity condition on the graph induced by these labellings to characterise the set of social choice functions implementable by an indirect mechanism.

We use this characterisation to show two settings in which the class of social choice functions implementable in extensive-form is the same as the class of social choice functions implementable by direct-revelation mechanisms. The first setting is that of single-parameter agents, where each agent's private information is expressible as a single number. Here we can separate an agent's type from her allocation and show that an allocation function is implementable by an indirect mechanism if and only if for each agent we can find a monotone restriction (with respect to the agent's bid) of the allocation function over all allocations it returns. This generalises the characterisation of [Archbold *et al.*, 2023] for single-parameter agents and we show that in this setting the class of allocation functions implementable by direct and indirect mechanisms are the same.

Main Result 1 (informal). *For any NOM indirect mechanism for single-parameter agents implementing a social choice function f there is a NOM direct mechanism implementing f.*

Therefore the implementation details do not matter when designing NOM mechanisms for single-parameter agents and we may restrict focus to direct mechanisms without loss of generality. In our second setting we consider mechanisms for two agents where each agent can have an arbitrary type but one of the agents has a "two-type" domain.

Main Result 2 (informal). *When there are two agents, one of which having only two possible types, for any NOM indirect mechanism there is an equivalent NOM direct mechanism.*

In this setting we show that in order for the revelation principle not to hold there must be a rich set from which we can choose the labelled profiles.

2 Preliminaries

There is a set $[n] = \{1, 2, \ldots, n\}$ of n *agents*. Each agent possesses some private information t_i known as her *type* that comes from some *domain* D_i of possible types. We denote by $D = \times_{i\in[n]} D_i$ the set of *type profiles*. For a type profile $t = (t_1, t_2, \ldots, t_n)$, agent i, and type $b_i \in D_i$, we adopt the standard notation (b_i, t_{-i}) to denote the type profile obtained from t by replacing t_i with b_i.

A *social choice function* $f : D \to O$ maps type profiles to outcomes; a mechanism is a process for first eliciting a type profile from the agents and then returning an outcome (plus a vector of payments). We are interested in designing indirect (otherwise known as extensive-form) mechanisms that implement some given social choice function. An *indirect mechanism* is a tuple $M = (f, p, T)$ where f is a social choice function, $p : D \to \mathbb{R}^n$ is a payment rule, and T is an *implementation tree* which describes the execution of the mechanism. Such a mechanism queries the agents about their types and finally select an outcome and payment vector based on their responses.

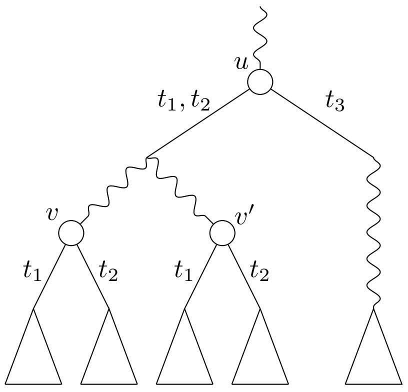

Figure 1: An extensive-form mechanism where player i has domain $D_i = \{t_1, t_2, t_3\}$ and is queried at u, v, and v'. The wedges represent the set of type profiles compatible with i signalling these types at various nodes in the tree.

The implementation tree describes how the agents are queried; formally, it is a rooted tree T with the following structure [Ferraioli *et al.*, 2022]. At each node u the mechanism queries exactly one player, denoted $S(u)$. Associated with node u is a set of type profiles *compatible* with u, denoted $D^u = (D^u_i, D^u_{-i})$. At the root node r of T we have $D^r = D$. The set of profiles compatible with the children of u partitions D^u. An action by player $i = S(u)$ at node u corresponds to choosing a child node of u. When v is the child node of u selected by player i, we say that i plays an action *compatible with* $b_i \in D^u_i$ whenever $b_i \in D^v_i$. We also say that i *signals* b_i at u. Each leaf ℓ of the tree is compatible with at least one profile $b \in D^\ell$, and once the mechanism reaches this leaf it immediately returns the outcome $M(b) = (f(b), p(b))$. We say that $i = S(u)$ *separates* two types $b_i, b'_i \in D^u_i$ at node u if $b_i \in D^v_i$ and $b'_i \in D^{v'}_i$ for two children v, v' of u in the implementation tree. We will denote by $D^u(t_j) = (D^u_i(t_j), D^u_{-i}(t_j))$ the set of all type profiles which f may elicit taken over all possible continuations of the players in the mechanism. In other words if the mechanism proceeds to node v from node u after player $i = S(u)$ signals type t_j then $D^u(t_j) = D^v$ is the set of all type profiles compatible with v. Note that i may still be queried again at some later point in the implementation tree.

Figure 1 provides an example implementation tree of some extensive-form mechanism for illustrative purposes, though we note different mechanisms may differ greatly in the structure of their implementation trees.

When mechanism M arrives at a leaf node ℓ of the implementation tree, we denote the outcome returned as $M(b) = (o, \boldsymbol{p}) \in O \times \mathbb{R}^n$, where $o = f(b)$, $\boldsymbol{p} = p(b)$, and $b \in D^\ell$. In other words, $M(b)$ is the output of the extensive-form mechanism when at each node each player i selects an action compatible with b_i. We consider each agent i's type as a function $t_i : O \to \mathbb{R}$ that maps each outcome of the mechanism to a real number. Agent utilities are quasi-linear, so on output $M(b) = (o, \boldsymbol{p})$ agent i receives utility $u_i(o, \boldsymbol{p}; t_i) = t_i(o) - p_i$. We overload the notation and write this utility as $t_i(o, \boldsymbol{p})$.

An agent's type is private information meaning they may lie when queried about their type if beneficial. [Troyan and Morrill, 2020] introduce non-obvious manipulability for direct-revelation mechanisms as a solution concept for agents with bounded rationality that only requires comparing the extremes of each agent's utility function under truthtelling and dishonesty. A mechanism M is *not obviously manipulable (NOM)* if the following two properties hold:

$$\sup_{b_{-i}}\{t_i(M(t_i, b_{-i}))\} \geq \sup_{b_{-i}}\{t_i(M(b_i, b_{-i}))\}, \qquad (1)$$

$$\inf_{b_{-i}}\{t_i(M(t_i, b_{-i}))\} \geq \inf_{b_{-i}}\{t_i(M(b_i, b_{-i}))\}, \qquad (2)$$

for every $t_i, b_i \in D_i$ and every $i \in [n]$. If (1) holds then M is *best-case non-obviously manipulable (BNOM)* and if (2) holds then M is *worst-case non-obviously manipulable (WNOM)*. If either inequality does not hold for a pair of types t_i, b_i then b_i is called an *obvious manipulation* of M.

For an extensive-form mechanism to be NOM then at every point at which the mechanism queries an agent about her type, the best- and worst-case outcomes from signalling her type truthfully must be no worse than the best- and worst-case outcomes from signalling a false type. Our definition of the implementation tree allows agents to signal different types each time they are queried by the mechanism. It is standard to assume, however, that agents commit to signalling a single type, or that the agent plays a *consistent strategy*. Otherwise, they play an *inconsistent strategy*. An indirect mechanism M is *NOM under inconsistent strategies* if for all i and all query nodes u such that $S(u) = i$ and for all $t_j, t_k \in D_i^u$:

$$\underset{b \in D_i^u(t_j)}{\text{ext}}\{t_j(M(b))\} \geq \underset{b' \in D_i^u(t_k)}{\text{ext}}\{t_j(M(b'))\}, \qquad (3)$$

for ext $\in \{\sup, \inf\}$. Similarly M is *NOM under consistent strategies* if for all i and all query nodes u such that $S(u) = i$ and for all $t_j, t_k \in D_i^u$:

$$\underset{b_{-i} \in D_{-i}^u(t_j)}{\text{ext}}\{t_j(M(t_j, b_{-i}))\} \geq \underset{b_{-i} \in D_{-i}^u(t_k)}{\text{ext}}\{t_j(M(t_k, b_{-i}))\}, \qquad (4)$$

for ext $\in \{\sup, \inf\}$. As usual if (3) or (4) holds for ext $= \sup$ then we will say M is *BNOM under (in)consistent strategies*, and if they hold for ext $= \inf$ then we will say M is *WNOM under (in)consistent strategies*. Observation 1 shows that assuming agents play a consistent strategy is without loss of generality.

In this paper we study implementability for indirect mechanisms. Social choice function f is *NOM-implementable (in extensive-form)* if there exists an indirect mechanism $M = (f, p, T)$ such that (4) holds for ext $\in \{\sup, \inf\}$. Since assuming consistent strategies is without loss of generality in the interest of space we will omit the definition for implementability under inconsistent strategies. BNOM- and WNOM-implementability are defined as expected.

3 Extensive Form Labellings

We study how to implement some given social choice function f with an extensive-form NOM mechanism. We may design both the payment rule and the implementation tree such that (1) and (2) are satisfied for every pair of types $t_i, b_i \in D_i$ and every agent $i \in [n]$. These conditions require comparing only the extremes (i.e., suprema and infima) of i's utility function for truthtelling versus dishonesty. As the mechanism designer we control the payments hence we effectively decide which bid profiles lead to these extremes. This is formalised by [Archbold *et al.*, 2023] in the form of profile labellings for direct mechanisms, and here we extend the concept to indirect mechanisms. These allow us to choose which bid profiles to compare for incentive-compatibility (plus some additional constraints imposed by the labelling) and allows us to characterise the set of implementable social choice functions.

Fix player i with domain D_i and let $d = |D_i|$. For each query node u such that $S(u) = i$, for every pair of separable types $t_j, t_k \in D_i^u$ at node u we introduce the *best-case* and *worst-case labellings* $\beta_{jk}^u, \omega_{jk}^u \in D_{-i}^u(t_k)$ representing the joint profile of the bidders excluding i that leads to i's best-case and worst-case (respectively) outcome when she has type t_j and signals type t_k at node u. These extremes of i's utility function are taken over all profiles in $D^u(t_k)$. These labellings induce the two following types of constraint. For each true type t_j and bid t_k of agent i at node u the *labelling constraints (for player i at node u)* ensure the validity of the labels with respect to i's true type:

$$t_j(M(\beta_{jk}^u)) \geq t_j(M(b)) \quad \text{for all } b \in D^u(t_k), \qquad (5)$$

$$t_j(M(\omega_{jk}^u)) \leq t_j(M(b)) \quad \text{for all } b \in D^u(t_k). \qquad (6)$$

The second type of constraint ensures the NOM incentive-compatibility constraints are met for each pair of separable types $t_j, t_k \in D_i^u$ at node u. The *incentive-compatibility constraints (for player i at node u)* ensure the extreme outcomes from signalling her type truthfully at node u are no worse than the extreme outcomes from signalling dishonestly:

$$t_j(M(\lambda_{jj}^u)) \geq t_j(M(\lambda_{jk}^u)) \quad \text{for all } t_j, t_k \in D_i^u \qquad (7)$$

for $\lambda^u \in \{\beta^u, \omega^u\}$. We denote by $\lambda = \{\lambda^u\}_{u:S(u)=i}$ the "full" labelling of the indirect mechanism for player i. With the following claim we show that allowing agents to play an inconsistent strategy does not alter the constraints semantically.

Observation 1. *It is without loss of generality for extensive-form NOM mechanisms to consider labellings $\lambda \in \{\beta, \omega\}$ in which player i plays a consistent strategy.*

Proof. For simplicity we give the proof based on Figure 1 but emphasise that the arguments can be applied to any tree. At node u player $i = S(u)$ separates types t_1 and t_2 from t_3. If she signals t_3 then this is the last time she is queried by the mechanism, while if she signals t_1 or t_2 at u then the mechanism will eventually progress to node v or v', depending on how the other players respond to their queries. In any case, at nodes v and v' player i separates t_1 from t_2 and these are the last times i is queried in the left hand subtree of T. We will argue that for any best-case labelling the assumption that i plays a consistent strategy does not semantically alter the constraints required for BNOM, and analogous reasoning will apply for any worst-case labelling ω and WNOM.

Due to the structure of T we must designate the best-case profiles β^u_{jk} for $j,k \in \{1,2,3\}$ and $\beta^v_{jk}, \beta^{v'}_{jk}$ for $j,k \in \{1,2\}$. Note that for any node where i separates two types then all leaves resulting from i signalling t_j must have the form (t_j, b_{-i}) for b_{-i} being some bid profile of the other players compatible with said node. Therefore β^u_{j3} must be of the form $(t_3,\cdot)$ for $j \in \{1,2,3\}$. Similarly for $j \in \{1,2\}$ the profiles $\beta^v_{j1}, \beta^{v'}_{j1}$ must be of the form $(t_1,\cdot)$ and $\beta^{v'}_{j2}$ must be of the form $(t_2,\cdot)$. Since t_1 and t_2 are inseparable at u then i may only play an inconsistent strategy if she signals one of these types at u. We will show that while enforcing consistent strategies will alter the constraints, it will only do so on an artificial level since the outcomes they encode will be the same. Fix i's true type to be t_1 and consider β^u_{11} of the form $(t_2,\cdot)$. The labelling constraints at u ensure that $t_1(M(\beta^u_{11}))$ is the maximum among all bid profiles in the set $D^u(t_1)$, including β^v_{11} and β^v_{12}. Now since β^u_{11} and β^v_{12} both appear in the same subtree of T then in addition to the labelling constraint at u enforcing $t_1(M(\beta^u_{11})) \geq t_1(M(\beta^v_{12}))$ there is the labelling constraint at v stating $t_1(M(\beta^v_{12})) \geq t_1(M(\beta^u_{11}))$, hence the outcomes of f at β^u_{11} and β^v_{12} are equal. Now by the incentive compatibility constraints at v we have $t_1(M(\beta^v_{11})) \geq t_1(M(\beta^v_{12})) = t_1(M(\beta^u_{11}))$. Since there is a labelling constraint at u enforcing $t_1(M(\beta^u_{11})) \geq t_1(M(\beta^v_{11}))$ then we have $\beta^u_{11} = \beta^v_{11} = \beta^v_{12}$.

Now consider i playing a consistent strategy where the labelled node, denoted $\hat{\beta}^u_{11}$, must be of the form $(t_1,\cdot)$. Using the same argument as before we get $\hat{\beta}^u_{11} = \beta^v_{11}$, since both profiles appear in the same subtree of T. Since we only change β^u_{11} to $\hat{\beta}^u_{11}$ then we have $\beta^u_{11} = \beta^v_{11} = \beta^v_{12}$ and therefore $\beta^u_{11} = \hat{\beta}^u_{11}$. As a final step consider the incentive compatibility constraints regardless of whether we assume consistent strategies, which require $t_1(M(\beta^u_{11})) \geq t_1(M(\beta^u_{13}))$ and $t_1(M(\hat{\beta}^u_{11})) \geq t_1(M(\beta^u_{13}))$. Since $\beta^u_{11} = \hat{\beta}^u_{11}$ then these constraints are exactly identical. Therefore enforcing consistent strategies does not alter any constraints semantically. $\square$

Thus we may assume agents play consistent strategies without loss of generality. We use the extensive-form labellings to characterise the social choice functions implementable by indirect mechanisms. We first use the labellings to define a particular graph for each agent and then use the cyclic monotonicity property to determine whether the mechanism coupled with such a labelling admits NOM payments. This technique was first used in [Rochet, 1987] to characterise truthfully implementable decision rules in arbitrary domains, and then it has been extended to characterise truthful implementability in a variety of settings, including functions on "rich" single-parameter domains [Bikhchandani *et al.*, 2006], on convex domains with finite range [Saks and Yu, 2005], and multiparameter settings for scheduling [Lavi and Swamy, 2009], verification [Ventre, 2014], and randomised mechanisms [Babaioff *et al.*, 2015].

Informally for some social choice function f extended with labelling λ for player i we will define a weighted multigraph $G^\lambda_{i,f}$ that represents each of the constraints required by some NOM mechanism (f,p). The edges of $G^\lambda_{i,f}$ will encode the incentive-compatibility constraints (7) and the labelling constraints (either (5) or (6), depending on whether λ is a best-case or worst-case labelling). Edge weights reflect how profitable it is for i to signal one type over another when queried at some node in the implementation tree.

More concretely $G^\lambda_{i,f} = (V, E^\lambda, w)$ is a weighted multigraph in which node set $V = D$ corresponds to the domain of social choice function f and edge set E^λ encodes the NOM constraints, whose precise definition we will give below. Since $G^\lambda_{i,f}$ is a multigraph we distinguish between edges connecting the same pair of nodes using a *type annotation* describing i's true type. We denote an edge between two nodes $x, y \in D$ annotated with type $t_i \in D_i$ as $(x,y;t)$ or equivalently $x \rightarrow^t y$. The weight of edge is $w(x,y;t) = t(f(x)) - t(f(y))$. Note that this definition is compatible with arbitrary domains and social choice functions.

Regardless of whether λ is a best- or worst-case labelling $G^\lambda_{i,f}$ encodes the incentive-compatibility constraints of the eventual mechanism, hence for $\lambda \in \{\beta, \omega\}$ and every node u such that $S(u) = i$ we have the edges $\{\lambda^u_{jj} \rightarrow^{t_j} \lambda^u_{jk} : t_j, t_k \in D\}$. For a best-case labelling β and worst-case labelling ω and every node u such that $S(u) = i$ we define the edges $\{\beta^u_{jk} \rightarrow^{t_j} \boldsymbol{b} : t_j, t_k \in D^u_i, \boldsymbol{b} \in D^u(t_k)\}$ and $\{\boldsymbol{b} \rightarrow^{t_j} \omega^u_{jk} : t_j, t_k \in D^u_i, \boldsymbol{b} \in D^u(t_k)\}$ encoding (5) and (6), respectively. The edge set E^λ for labelling λ is simply the union of the incentive-compatibility edges with the appropriate labelling edges. We now use cycle monotonicity on this graph to characterise the class of implementable social choice functions.

Theorem 1. *Social choice function f with implementation tree T is BNOM-implementable in extensive-form (respectively, WNOM-implementable in extensive-form) if and only if there exists a best-case labelling β (respectively, worst-case labelling ω) of f for player i such that $G^\beta_{i,f}$ (respectively, $G^\omega_{i,f}$) has no negative cycles, for all players $i \in [n]$.*

Proof. ($\implies$) By the implementability of f we may therefore define a payment rule p such that all labelling and incentive-compatibility constraints are satisfied for the given labelling of f with implementation T. For the sake of contradiction suppose that every labelling of f with tree T for player i induces a negative cycle in the graph $G^\lambda_{i,f}$. Let $C = \langle x^{(1)} \rightarrow^{t_1} x^{(2)} \rightarrow^{t_2} \ldots \rightarrow^{t_{m-1}} x^{(m)} \rightarrow^{t_m} x^{(1)} \rangle$ be such a cycle between any m nodes with negative weight. Since (f,p,T) defines a NOM mechanism then each of the constraints represented by the edges of C are satisfied, i.e., $t_k(f(x^{(k)})) - p_i(x^{(k)}) \geq t_k(f(x^{(k+1)})) - p_i(x^{(k+1)})$ for $k \in [m]$ where index $m+1$ wraps around to 1. Summing these inequalities yields $t_1(f(x^{(1)})) - t_1(f(x^{(2)})) + t_2(f(x^{(2)})) - t_2(f(x^{(3)})) + \ldots + t_m(f(x^{(m)})) - t_m(f(x^{(1)})) \geq 0$, which is exactly the expression for the weight of cycle C. Hence we have a contradiction and there must be a labelling λ of f

with implementation T such that all cycles in $G^{\lambda}_{i,f}$ are non-negative.

($\Longleftarrow$) Let λ be a profile labelling of f implemented by T such that the induced graph $G^{\lambda}_{i,f}$ has no negative weight cycles. We will define payments such that (f, p, T) is a NOM indirect mechanism as follows. First augment $G^{\lambda}_{i,f}$ by introducing an artificial node γ and edges $(\gamma, x; 0)$ for every node $x \in V$, where 0 is an artificial type that assigns 0 to every outcome of f and hence these edges will have zero weight. Call this augmented graph $\mathcal{G}^{\lambda}_{i,f}$. Note that the addition of these edges does not create any new cycles we have only introduced outgoing edges from γ and since there are no negative cycles in $G^{\lambda}_{i,f}$ then shortest paths are well-defined in $\mathcal{G}^{\lambda}_{i,f}$. We now set payment $p_i(x) = \mathrm{SP}(\gamma, x)$, where $\mathrm{SP}(x, y)$ denotes the shortest path between nodes x and y in $\mathcal{G}^{\lambda}_{i,f}$. By definition of shortest paths for any edge $(x, y; t)$ in $\mathcal{G}^{\lambda}_{i,f}$ we must have $\mathrm{SP}(\gamma, y) \leq \mathrm{SP}(\gamma, x) + w(x, y; t)$, and therefore for any such edge we have $t(f(y)) - p_i(y) \leq t(f(x)) - p_i(x)$.

It is straightforward to show that whenever such an edge $(x, y; t)$ is present in the graph the corresponding constraint is satisfied with these definitions of payments: either $t = x_i$ and we have either a (truthful) labelling constraint or an incentive compatibility constraint (depending on the labelling λ), or $t \neq x_i$ and we have a (dishonest) labelling constraint. Moreover it suffices to only consider two nodes separated at some lowest common ancestor in the tree: if the label for some action for parent node u and child node v is different then as a consequence of Observation 1 the utility derived by i is the same in both outcomes associated with these nodes. $\square$

In the next section we use this result to characterise the functions implementable by an indirect mechanism for single-parameter agents and show these is no different to those implementable by a direct mechanism.

4 The Revelation Principle for Single-Parameter Agents

A single-parameter agent is one whose type can be described by a single number: when agent i has type t_i then she values the outcome of the allocation function $f(b)$ as the product $t_i \cdot f_i(b)$ of her private type and her personal allocation. This allows us to decouple player types from the output of a mechanism's allocation function to prove structural properties about the latter in isolation. [Archbold *et al.*, 2023] show that an allocation function f is implementable by a direct-revelation NOM mechanism if and only if it is *overlapping*. If f is overlapping then we may always find a "monotone restriction" of f when viewed as a (multi-valued) allocation function of player i's bid. In other words, there is always a set of profiles ensuring that i is allocated more as she bids more. We generalise this notion to show a similar condition for implementability by indirect mechanisms, and show that the functions implementable by indirect mechanisms is no different than those implementable by direct ones.

Let u be some node of the implementation tree T for allocation function f with $S(u) = i$. We denote by $O^u_{ij} = \{f_i(t_j, b_{-i}) : b_{-i} \in D^u_{-i}(t_j)\}$ the set of all outcomes of f to player i when i signals type t_j at u, taken over all possible continuations of the mechanism from the other players. Furthermore let $O^u_i = \times_{t_j \in D_i} O_{ij}$. We say that f is *overlapping for i at node u* with $i = S(u)$ if there exists a non-decreasing vector $(o_1, o_2, \dots, o_d) \in O^u_i$. This says that when i is responding to the mechanism at node u there always exists an outcome of f such that when i has a larger type she is allocated more by the mechanism. Notice that if two types t_j, t_k are indistinguishable at node u then the ranges of f from signalling either type are the same. We characterise the class of social choice functions NOM-implementable by an indirect mechanism with the following.

Lemma 1. *Social choice function f is NOM-implementable by an indirect mechanism if and only if there exists some implementation tree T such that f is overlapping at every node u with $S(u) = i$, for each player $i \in [n]$.*

The proof employs the same arguments as [Archbold *et al.*, 2023] to show both directions. Since the social choice function f we wish to implement is given to us then the range of outcomes $\{f_i(b_i, b_{-i})\}_{b_{-i}}$ of the function to player i for each bid b_i is fixed. The extra flexibility provided by an indirect mechanism over a direct-revelation implementation of f is that we may choose how to partition this set of outcomes for a given bid using the nodes of the implementation tree. The sufficiency follows from the fact that, given f which is overlapping at u for some implementation, then we can always define an extensive-form labelling ensuring no negative cycles in the graph $G^{\lambda}_{i,f}$. For necessity, supposing that f is not overlapping at u leads to negative cycles in $G^{\lambda}_{i,f}$ and hence contradiction by Theorem 1.

We now wish to show that the set of NOM-implementable allocation functions for direct and indirect mechanisms is the same for single-parameter agents. Our definition of an indirect mechanism is able to emulate a direct revelation mechanism: simply query each player once and get them to separate every type in their domain from one another. This amounts to asking each player to directly declare their type to the mechanism. Therefore all direct-revelation NOM-implementable allocation functions are implementable by an indirect mechanism. This of course covers cases where agents are single-parameter. We may use Lemma 1 to show the opposite: for single-parameter agents all NOM-implementable functions by an indirect mechanism are implementable by a direct mechanism. The functions implementable by direct and indirect mechanisms is therefore the same, thus the revelation principle holds for single-parameter agents.

Theorem 2. *For every NOM indirect mechanism implementing social choice function f for single-parameter agents there is a NOM direct mechanism implementing f.*

Proof. Let f be a NOM-implementable social choice function that is implemented by indirect mechanism (f, p, T). Note that by Lemma 1 f must be overlapping for $S(u)$ at each node u of T. We will show that restructuring the implementation to one in which player i is queried once at the root and fully separates each type in her domain, which must occur in a direct-revelation implementation of f, maintains the overlapping property of f at the root. Therefore the resulting implementation will also be NOM.

First recall the definition of O^u_{ij}, which denotes the set of all outcomes possible when player i signals t_j at node u. Note that for some player $i = S(u)$ and two types $t_j, t_k \in D^u_i$ that are overlapping at node u, the overlapping property of f at u may only be violated if we restrict either of the two sets O^u_{ij} and O^u_{ik}. Now fix a player i for which there are two types $t_j, t_k \in D^v_i$ which are inseparable at u but separable at child node v of u. Clearly the set of all outcomes reachable after signalling t_1 at u contains all outcomes reachable after signalling t_1 at v, hence $O^v_{ij} \subseteq O^u_{ij}$. Thus modifying the tree T such that i now separates t_j from t_k at node u rather than node v cannot violate the overlapping property of f at u (which was true by hypothesis). We may repeat this argument and continue to query i further up the tree, where she separates progressively more types from her domain. Our final modified tree T' will query i once at the root, at which point she separates each type from every other. At each point the overlapping property is maintained and thus f is also overlapping for i at the root. Hence f is implementable by this modified tree T' by Lemma 1.

As a final step we will use the overlapping property of [Archbold *et al.*, 2023] to show f is NOM-implementable by a direct mechanism. Note that in a direct mechanism each player is "queried" exactly once simultaneously and directly reports her type, effectively separating all types in her domain. We have reconstructed T into T' such that i is now queried once at the root and fully separates each type in her domain. Hence from i's perspective T' is a direct-revelation implementation of f and since f is overlapping for i at the root in implementation T' so too is f overlapping for i in the sense of [Archbold *et al.*, 2023]. Note that we do not need to move each player to the root simultaneously – to show the existence of a direct-revelation mechanism it suffices to show that this can be done for each player in isolation while maintaining the overlapping property. Thus there must exist a direct-revelation mechanism that NOM-implements f. □

5 Beyond Single-Parameter Domains

In this section we prove the revelation principle for a setup that is independent of agents' domains. This result says that for the revelation principle not to hold there must be a rich set of labels that the extensive-form mechanism can use. A *two-type domain* is a domain containing two possible types, where the types themselves can be arbitrary.

Theorem 3. *The revelation principle holds for NOM mechanisms when $n = 2$ and one of the two agents has a two-type domain.*

Proof. Assume without loss of generality that agent 2 has a two-value domain, $D_2 = \{t, t'\}$. Fix the social choice function f; we will prove that if there is an extensive-form NOM mechanism M that implements f then there is a direct-revelation mechanism M' that is NOM and also implements f. Let $G^\lambda_{i,f}$ and $\mathcal{G}^\ell_{i,f}$ be the graphs of interest for extensive-form and direct-revelation labellings λ and ℓ, respectively.

We start by observing that the revelation principle holds true for agent 2. Assume by contradiction that this is not true. This means that for any ℓ there is a negative-weight two-cycle in $\mathcal{G}^\ell_{i,f}$ between $(\ell_{tt'}, t)$ and $(\ell_{t't}, t')$ and then the outcome of the mechanism for agent 2 must differ in these two profiles (for otherwise, each edge would weigh 0). Therefore, to implement the mechanism, the extensive-form implementation must separate these profiles. Clearly, this cannot be agent 2 or we would get the same negative cycle. Thus, agent 1 must separate $\ell_{tt'}$ from $\ell_{t't}$ before agent 2 separates t from t'. Reiterating the argument for all the possible labellings ℓ yields the contradiction that agent 2 can never separate the two types in her domain.

Let us now focus on the revelation principle for agent 1. Since agent 2 has a two-value domain, then any extensive-form mechanism, including M, has only one node v in each path from the root to a leaf of its implementation tree T such that $S(v) = 2$. Thus, the only difference between a direct-revelation mechanism M' and an extensive-form mechanism M for agent 1 is that the latter restricts the labels λ^u at nodes u below v to belong to D^u_2 whereas the domain of the labels would be unrestricted for the former. Labels are unrestricted for both M and M' at nodes above v.

Therefore, if the children of v are leaves of T then M has the same labels (and then graphs) of M', thus proving the revelation principle. Otherwise, let us focus on a type $t_1 \in D^v_1$, i.e., t_1 is still in the domain of agent 1 when 2 is queried at v. Since $|D_2| = 2$ then there are going to be labels λ^u for t_1 at ancestors u of v and labels λ^z for t_1 at descendants z of v that are equal. Recall that λ^u would be a feasible choice of a label $\ell_{t_1\star}$ for a direct-revelation mechanism M'. By inspection, this means that $G^\lambda_{i,f}$ contains $\mathcal{G}^\ell_{i,f}$ as a sub-graph. Since M is NOM then both $G^\lambda_{i,f}$ and $\mathcal{G}^\ell_{i,f}$ have no negative-weight cycles, thus proving that M' is NOM. □

6 Conclusion

In this work we extend the work on studying the functions which can be implemented as mechanisms that are not obviously manipulable. We formalise the constraints required by incentive compatibility in these settings using the concept of extensive-form profile labellings and use cycle monotonicity to characterise the social choice functions that admit NOM payments. Applying this to mechanisms for single-parameter agents we find that we may restrict our attention to the simpler direct mechanisms without sacrificing incentive properties, and similarly for the setting of two players in which one has a two-value domain. An interesting direction would be to study what leverage indirect mechanisms for single-parameter agents *do* provide over direct mechanisms with respect to other desirable notions, e.g. fairness. It would also be useful to further explore the relationship between direct-revelation and extensive-form NOM mechanisms for other agent domains beyond those covered by Theorems 2 and 3.

Acknowledgements

Carmine Ventre acknowledges funding from the UKRI Trustworthy Autonomous Systems Hub (EP/V00784X/1).

References

[Archbold *et al.*, 2023] Thomas Archbold, Bart De Keijzer, and Carmine Ventre. Non-obvious manipulability for single-parameter agents and bilateral trade. In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023)*, January 2023.

[Aziz and Lam, 2021] Haris Aziz and Alexander Lam. Obvious manipulability of voting rules. In Dimitris Fotakis and David Ríos Insua, editors, *Algorithmic Decision Theory*, pages 179–193, Cham, 2021. Springer International Publishing.

[Babaioff *et al.*, 2015] Moshe Babaioff, Robert D. Kleinberg, and Aleksandrs Slivkins. Truthful mechanisms with implicit payment computation. *J. ACM*, 62(2), may 2015.

[Bade and Gonczarowski, 2017] Sophie Bade and Yannai A. Gonczarowski. Gibbard-satterthwaite success stories and obvious strategyproofness. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, EC '17, page 565, New York, NY, USA, 2017. Association for Computing Machinery.

[Bikhchandani *et al.*, 2006] Sushil Bikhchandani, Shurojit Chatterji, Ron Lavi, Ahuva Mu'alem, Noam Nisan, and Arunava Sen. Weak monotonicity characterizes deterministic dominant-strategy implementation. *Econometrica*, 74(4):1109–1132, 2006.

[de Keijzer *et al.*, 2020] Bart de Keijzer, Maria Kyropoulou, and Carmine Ventre. Obviously Strategyproof Single-Minded Combinatorial Auctions. In Artur Czumaj, Anuj Dawar, and Emanuela Merelli, editors, *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*, volume 168 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 71:1–71:17, Dagstuhl, Germany, 2020. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

[Dur *et al.*, 2018] Umut Dur, Robert G. Hammond, and Thayer Morrill. Identifying the harm of manipulable school-choice mechanisms. *American Economic Journal: Economic Policy*, 10(1):187–213, February 2018.

[Ferraioli and Ventre, 2021] Diodato Ferraioli and Carmine Ventre. Approximation guarantee of OSP mechanisms: The case of machine scheduling and facility location. *Algorithmica*, 83(2):695–725, 2021.

[Ferraioli *et al.*, 2019] Diodato Ferraioli, Adrian Meier, Paolo Penna, and Carmine Ventre. Obviously Strategyproof Mechanisms for Machine Scheduling. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms (ESA 2019)*, volume 144 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 46:1–46:15, Dagstuhl, Germany, 2019. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[Ferraioli *et al.*, 2022] Diodato Ferraioli, Paolo Penna, and Carmine Ventre. Two-way greedy: Algorithms for imperfect rationality. In Michal Feldman, Hu Fu, and Inbal Talgam-Cohen, editors, *Web and Internet Economics*, pages 3–21, Cham, 2022. Springer International Publishing.

[Kiedanski *et al.*, 2020] Diego Kiedanski, Ariel Orda, and Daniel Kofman. Combflex: a linear combinatorial auction for local energy markets. In *2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*, pages 1–7, 2020.

[Lavi and Swamy, 2009] Ron Lavi and Chaitanya Swamy. Truthful mechanism design for multidimensional scheduling via cycle monotonicity. *Games and Economic Behavior*, 67(1):99–124, September 2009.

[Li, 2017] Shengwu Li. Obviously Strategy-Proof Mechanisms. *American Economic Review*, 107(11):3257–3287, November 2017.

[Nisan *et al.*, 2007] Noam Nisan, Tim Roughgarden, Éva Tardos, and Vijay V. Vazirani. Algorithmic game theory. Cambridge University Press, 2007.

[Ortega and Segal-Halevi, 2022] Josué Ortega and Erel Segal-Halevi. Obvious manipulations in cake-cutting. *Soc. Choice Welf.*, 59(4):969–988, 2022.

[Psomas and Verma, 2022] Alexandros Psomas and Paritosh Verma. Fair and efficient allocations without obvious manipulations. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 13342–13354. Curran Associates, Inc., 2022.

[Rochet, 1987] Jean-Charles Rochet. A necessary and sufficient condition for rationalizability in a quasi-linear context. *Journal of Mathematical Economics*, 16:191–200, 1987.

[Roth, 1991] Alvin E. Roth. A natural experiment in the organization of entry-level labor markets: regional markets for new physicians and surgeons in the united kingdom. *The American economic review*, 81(3):415 – 440, 1991. Cited by: 209.

[Saks and Yu, 2005] Michael Saks and Lan Yu. Weak monotonicity suffices for truthfulness on convex domains. In *Proceedings of the 6th ACM Conference on Electronic Commerce*, EC '05, page 286–293, New York, NY, USA, 2005. Association for Computing Machinery.

[Troyan and Morrill, 2020] Peter Troyan and Thayer Morrill. Obvious manipulations. *Journal of Economic Theory*, 185, 2020. Article 104970.

[Troyan *et al.*, 2020] Peter Troyan, David Delacrétaz, and Andrew Kloosterman. Essentially stable matchings. *Games Econ. Behav.*, 120:370–390, 2020.

[Troyan, 2022] Peter Troyan. Non-Obvious Manipulability of the Rank-Minimizing Mechanism. Papers 2206.11359, arXiv.org, June 2022.

[Ventre, 2014] Carmine Ventre. Truthful optimization using mechanisms with verification. *Theoretical Computer Science*, 518:64–79, January 2014.