Implementation with Contingent Contracts *

Rahul Deb † and Debasis Mishra ‡

January 1, 2014

Abstract

We study dominant strategy incentive compatibility in a mechanism design setting with contingent contracts where the utility of each agent is observed by the principal and can be contracted upon. Our main focus is on the class of linear contracts (one of the most commonly observed contingent contracts) which consist of a transfer and a flat rate of profit sharing. We first demonstrate the applications of linear contracts. We show that they can achieve efficient outcomes with budget balance overcoming a known shortcoming of the VCG mechanism. Additionally, they can be used to implement social outcomes (like the Rawlsian) that are not incentive compatible using transfers alone. We then give implicit (using a condition called acyclity) and explicit characterizations of social choice functions that are implementable using linear contracts. Further, we provide a foundation for them by showing that, in finite type spaces, every social choice function that can be implemented using a more general nonlinear contingent contract can also be implemented using a linear contract.

*This paper is a significant revision of an earlier paper titled "Implementation with Securities". We are grateful to the co-editor Matthew Jackson and three anonymous referees for their insightful comments and suggestions on the earlier version of the paper. We are also grateful to Juan Carlos Carbajal, Arunava Sen, Andy Skrzypacz, Rakesh Vohra and numerous seminar audiences for valuable feedback.

†University of Toronto. Email: rahul.deb@utoronto.ca

‡Indian Statistical Institute, Delhi. Email: dmishra@isid.ac.in

1 Introduction

The classic setting in mechanism design with quasi-linear utilities is the following. Agents privately observe their types and make reports to the mechanism designer. Based on these reports, the mechanism designer chooses an alternative and transfer amounts. Agents then realize their utility from the chosen alternative and their final payoff is this utility less their transfer amount. We refer to such mechanisms as quasilinear mechanisms. An important aspect of this setting is that the mechanism is a function only of the reports and not of the realized utilities of the agents. This could either be because the principal cannot observe these utilities or that they are not verifiable by third parties and hence contracts based on them cannot be enforced.

However, in many practical settings, principals can and do offer contracts which are functions of both the agents' reports and their realized utilities. These are called *contingent contracts*. Perhaps the simplest and most commonly observed example of a contingent contract is a *linear contract*. Here the contract consists of an lump sum transfer and a flat percentage (such as a royalty rate or a tax) which determines how the principal and agent share the latter's utility. Contingent contracts are ubiquitous and they are commonly observed in the form of taxes used to finance public goods provision. Other settings where they are used include publishing agreements with authors, musicians seeking record labels, entrepreneurs selling their firms to acquirers or soliciting venture capital, and sports associations selling broadcasting rights. In addition, auctions are often conducted in which buyers bid using such contracts as opposed to simply making cash bids. Examples include the sale of private companies and divisions of public companies, government sales of oil leases, wireless spectrum and highway building contracts.

Here, like the standard mechanism design setting, the agents first report their types to the principal who then chooses an alternative based on these reports. The utility of an agent, which depends on his true type and the chosen alternative, is then realized. Unlike, the standard setting however, the realized utility from the chosen alternative is observable not just to the agent but also to the principal, and since it is contractible, the payoffs from the contract to both can depend on it. Additionally, there is often uncertainty in these environments. At the interim stage (after realizing the type but before the alternative is chosen), agents only know the distribution of utilities that arise from each alternative.

This paper is the first to study the problem of dominant strategy implementation in such a general environment. With this implementation criterion, it is not necessary to assume that either the principal or the agents have prior beliefs over the types of all agents. A mechanism in this context consists of a social choice function (scf) and a contingent contract which determines each agent's payoff as a function of their realized utility and the profile of

announced types. We say that an scf is implementable using a (linear) contingent contract if there exists a (linear) contingent contract such that truthful reporting of type is a dominant strategy for each agent in the resulting mechanism.

1.1 Summary of Results

We first demonstrate how expanding the set of contracts from quasilinear transfers to linear contingent contracts can be useful for achieving desirable social outcomes. These results can be grouped into the following two categories.

Efficiency with Budget Balance. A seminal impossibility result in mechanism design theory is that no efficient and dominant strategy quasi-linear mechanism can be budget balanced (Green and Laffont, 1979). An implication of this impossibility is that even though it is possible to implement the efficient scf (for instance, using the VCG mechanism), it is not possible to redistribute the resulting welfare. In contrast to this impossibility result, we show that there exist simple linear contracts that achieve efficiency and, in addition, are budget-balanced and individually rational. The mechanism we construct divides the social welfare of an efficient social choice function equally among all the agents.

Implementing Aggregate Utility Maximizers. We show that using linear contracts, the mechanism designer can implement a larger class of scfs than those possible using quasilinear transfers. We identify a family of scfs called *aggregate utility maximizers* (supplemented with a tie-breaking rule) and show that they are implementable using linear contracts if the type space is finite. An scf is an aggregate utility maximizer if it maximizes a social welfare function which depends on the agents' utilities and the alternative and is (weakly) increasing in the agents' utilities.

As an application of this result, consider a planner interested in the goal of reducing inequality (as opposed to efficiency). This is typically modeled using the max-min or Rawlsian scf in which the planner chooses an alternative that maximizes the minimum utility of agents. It has been shown that the Rawlsian scf may not be implementable using quasilinear transfers. However, since the Rawlsian scf is an aggregate utility maximizer, our result implies that it is implementable using a linear contract. Thus, linear contracts allow the principal to achieve certain important welfare objectives which may not be possible to implement using quasilinear transfers.

Having demonstrated these applications of linear contracts, we then provide a characterization of the set of scfs that are implementable.

Characterization of Implementable SCFs. We show that if the type space is finite, any scf implementable using a general nonlinear contingent contract can also be implemented using a linear contract. Put differently, this result states that the set of scfs implementable by linear contracts is not expanded by using contingent contracts that depend nonlinearly on the realized utility of the agents. This result can be interpreted as a foundation for linear contracts and provides one explanation for their ubiquity in practical applications.[1]

Further, we show that the set of scfs implementable by linear contracts is characterized by a condition called *acyclicity*, which is simple to interpret and apply. As an application, we show that (the above mentioned) aggregate utility maximizers are acyclic and are hence implementable. When the type space satisfies an additional richness condition, implementability is characterized by a weaker condition called *2-acyclicity*. Moreover, under this richness condition, we show that the an scf (which breaks ties consistently) is implementable only if it is an aggregate utility maximizer. Thus, under reasonable assumptions on the type space, we completely characterize the set of implementable scfs using linear contracts.

Finally, we show that (under a mild condition) all implementable scfs can be implemented using linear contracts that satisfy two appealing properties: they are individually rational and the agents make non-negative payments. Hence, the mechanism designer neither has to pump in additional sums of money nor has to break individual rationality for implementation.

1.2 Organization of the Paper

The rest of the paper is organized as follows. For ease of exposition, the bulk of the analysis in the paper is conducted in a simplified deterministic framework which is described in Section 2. We show the existence of an efficient, dominant strategy incentive compatible, budget-balanced, and individually rational linear mechanism in Section 3. In Section 4, we show that the family of aggregate utility maximizers can be implemented using linear contracts. In Section 5, we show the implementability equivalence between contingent and linear contracts, and provide implicit and explicit characterizations of scfs that are implementable using linear contracts. Then, we discuss how these results can be generalized to environments with uncertainty in Section 6. In order to make formal connections with our model, we defer the discussion of the related literature to Section 7. Finally, in Section 8, we discuss the extent to which some of the results can be extended and provide a few avenues for future research. Appendix 1 contains the proofs of the theorems which are missing from the body

[1] Though this result requires finiteness of type space, it can be showed to hold in an uncountable type space under additional technical conditions.

and Appendix 2 has some extensions of our results.

2 The Deterministic Model

There is a set of agents $N := \{1, \ldots, n\}$ who face a mechanism designer (principal). The set of alternatives is A. For ease of exposition, we begin by examining a deterministic model and the majority of the analysis in the paper will be conducted in this framework. Here, the type of an agent i is given by a map $v_i : A \to \mathbb{R}$ and V_i denotes the set of all possible types of agent i. Using the standard notation, $V := V_1 \times \ldots \times V_n$ denotes the set of types of all the agents and $V_{-i} := \prod_{j \neq i} V_j$ is the set of types of all agents except i. In this deterministic environment, the ex-post utility of agent i with type v_i for an alternative a is given by $v_i(a)$, and is observed by both the agent and the mechanism designer.[2] We assume that there are no two distinct types v_i, v_i' such that $v_i(a) = v_i'(a)$ for all $a \in A$ or, in words, that there are no two identical types with different names.[3]

In Section 6, we describe the general model with uncertainty. There, the ex-post utility of agent i is a random variable, the distribution of which depends on the type v_i and the alternative a. At the interim stage (that is, after the type is realized and before an alternative is chosen), this ex-post utility is not known to both the mechanism designer and the agents.

A **social choice function (scf)** is a map $f : V \to A$. This map specifies the chosen alternative for every reported profile of types.

The fundamental difference separating our model from the standard mechanism design setting is that the ex-post utility of every agent is *contractible*. A commonly observed contract which has this feature is a **linear contract**. A linear contract for agent i consists of two mappings, a royalty (or tax) rule $r_i : V \to (0, \infty)$ and a transfer rule $t_i : V \to \mathbb{R}$. A **linear mechanism** $(f, (r_1, t_1), \ldots, (r_n, t_n))$ consists of a linear contracts (r_i, t_i) for each agent i and an scf f. The payoff assigned to agent i by a linear mechanism is

$$r_i(v_i', v_{-i}')v_i(f(v_i', v_{-i}')) - t_i(v_i', v_{-i}'),$$

if his true type is v_i and the profile of reported types is (v_i', v_{-i}'). In words, a linear contract specifies a transfer amount and a fraction of the utility to be shared. Notice that we do not allow $r_i(v_i, v_{-i}) = 0$ for any profile of types v_i, v_{-i}. The main reason we impose this restriction is to prevent the principal from "buying" the agents, thereby making them indifferent amongst reports and trivializing the implementation problem.

[2] We use the term 'utility' to distinguish this from the final 'payoff' that the contract awards.

[3] This assumption is not necessary for the results and is made to reduce cumbersome notation and additional qualifiers in the statements of the theorems.

A special case of the linear mechanism is the standard **quasi-linear mechanism** $(f, t_1, \ldots, t_n)$, in which the contracts just specify transfers, and where $r_i(\cdot) = 1$ for all i. The payoff assigned to agent i by such a quasi-linear mechanism is $v_i(f(v_i', v_{-i}')) - t_i(v_i', v_{-i}')$ if the agent's true type is v_i and the profile of reported types is (v_i', v_{-i}').

An important aspect of linear contracts is that the payoff awarded by the contract is increasing in the realized utility $v_i(\cdot)$ of the agent since the r_i's are restricted to being positive. We now define a general nonlinear class of contracts which satisfy this property. A **contingent contract** of agent i is a map $s_i : \mathbb{R} \times V \to \mathbb{R}$ which is strictly increasing in the first argument. A contingent contract of agent i assigns a payoff to him for every realized ex-post utility and for every profile of reported types. A **contingent mechanism** is $(f, s_1, \ldots, s_n)$, where f is an scf and $(s_1, \ldots, s_n)$ are the contingent contracts of the agents. The payoff assigned to agent i by a contingent mechanism is

$$s_i(v_i(f(v_i', v_{-i}')), v_i', v_{-i}'),$$

if his true type is v_i and the profile of reported types is (v_i', v_{-i}'). Note that, since s_i is strictly increasing in the first argument, the assigned payoff by a contingent contract is strictly larger for greater realized utilities. Notice also that a linear contract is a special case of a contingent contract.

The timing of the model can be summarized as follows:

Types v_i are realized	$\longrightarrow$	Types v_i' reported to principal	$\longrightarrow$	Principal chooses alternative a	$\longrightarrow$	Agents realize utility $v_i(a)$	$\longrightarrow$	Contract offers payoffs as a function of $v_i(a)$ and (v_i', v_{-i}')

While the contingent contracts we consider are very general and model many real world contracts, they are with loss of generality. Requiring s_i to be strictly increasing in the first argument is not completely innocuous as it rules out certain commonly used contracts which are weakly increasing such as call options and convertible debt. Again, this assumption is made is to prevent the principal from making agents indifferent amongst reports (for instance, by buying the agents). Additionally, notice that we do not allow the payoff to agent i from the contingent contract to depend on the realized utilities of the other agents but only on their announced types. This is true in most real world contingent contracts and, to the best of our knowledge, this simplifying assumption is made in all of the papers in the literature.

Most importantly, in this deterministic version of our framework, the monotonicity restriction may prevent the principal from punishing detectable misreports from the agent.

Here, the realized utility may reveal the true type of the agent and thus, in principle, contracts can be written which impose large punishments whenever misreports are detected. Such punishments may not be possible using a contingent contract as the monotonicity requirement will then impose a restriction on the payoffs that the contract can offer other agents. That said, we should point out that this deterministic version of our model is merely for expositional purposes and in the general version of our model with uncertainty (described in Section 6), realized utilities do not generally reveal types.

We now define the notion of dominant strategy implementation that we use.

DEFINITION **1** *An scf f is* **implementable by a linear contract** *in dominant strategies if there exist linear contracts $((r_1, t_1), \ldots, (r_n, t_n))$ such that for every $i \in N$, for every $v_{-i} \in V_{-i}$, we have*

$$r_i(v_i, v_{-i})v_i(f(v_i, v_{-i})) - t_i(v_i, v_{-i}) \geq r_i(v_i', v_{-i})v_i(f(v_i', v_{-i})) - t_i(v_i', v_{-i}) \qquad \forall\ v_i, v_i' \in V_i.$$

In this case, we say that the linear mechanism $(f, (r_1, t_1), \ldots, (r_n, t_n))$ is incentive compatible.

The notion of implementation with contingent contracts can be defined analogously.

DEFINITION **2** *An scf f is* **implementable by a contingent contract** *in dominant strategies if there exist contingent contracts $(s_1, \ldots, s_n)$ such that for every $i \in N$, for every $v_{-i} \in V_{-i}$, we have*

$$s_i(v_i(f(v_i, v_{-i})), v_i, v_{-i}) \geq s_i(v_i(f(v_i', v_{-i})), v_i', v_{-i}) \qquad \forall\ v_i, v_i' \in V_i.$$

In this case, we say that the contingent mechanism $(f, s_1, \ldots, s_n)$ is incentive compatible.

We will be concerned with two important properties of mechanisms which are defined below.

DEFINITION **3** *A contingent mechanism $(f, s_1, \ldots, s_n)$ is* **budget-balanced** *if at every type profile $v \equiv (v_1, \ldots, v_n) \in V$, we have*

$$\sum_{i \in N} s_i(v_i(f(v)), v) = \sum_{i \in N} v_i(f(v)).$$

A contingent mechanism $(f, s_1, \ldots, s_n)$ is (ex-post) **individually rational** *if at every type profile $v \equiv (v_1, \ldots, v_n)$ and every $i \in N$, we have*

$$s_i(v_i(f(v)), v) \geq 0.$$

Budget-balance requires the sum of payoffs of the agents from the contingent contracts to be equal to the sum of ex-post utilities, that is, the designer does not add or take away any aggregate utilities of the agents. Individual rationality requires that the payoff of each agent from the contingent mechanism is non-negative.

3 Achieving Efficiency with Budget Balance

This is the first of two sections which provide applications of linear contracts. The aim is to demonstrate that these contracts are powerful tools that allow a planner to achieve objectives which are not possible with standard quasilinear mechanisms. In this section, we focus on efficient allocations.

An scf f^* is **efficient** if at every profile of types $v \equiv (v_1, \ldots, v_n) \in V$, we have

$$f^*(v) \in \underset{a \in A}{\operatorname{argmax}} \sum_{i \in N} v_i(a).$$

Additionally, a linear mechanism with such an scf is called efficient. A seminal result in the mechanism design literature is that the efficient scf can be implemented using Groves transfer rules (Groves, 1973). However, Green and Laffont (1979) showed that it is impossible to satisfy efficiency, dominant strategy incentive compatibility, and budget-balancedness using quasi-linear mechanisms. This impossibility can be overcome by considering a weaker notion of implementability - Bayesian implementation. Arrow (1979) and d'Aspremont and Gérard-Varet (1979) construct a Bayesian incentive compatible mechanism, known as the dAGV mechanism, that is efficient and budget-balanced. However, it is not individual rational. Indeed, Myerson and Satterthwaite (1983) formally establish that it is impossible to satisfy efficiency, Bayesian incentive compatibility, budget-balancedness, and individual rationality, even in a simple two agent model of bilateral trading.

We now define a linear contract which overcomes these impossibilities.

Definition 4 *An* **equal sharing** *mechanism consists of an efficient scf f^* and linear contracts $((r_1^*, t_1^*), \ldots, (r_n^*, t_n^*))$, where for every type profile $v \equiv (v_1, \ldots, v_n) \in V$ and for every $i \in N$*

$$r_i^*(v) := \frac{1}{n}$$
$$t_i^*(v) := -\frac{1}{n} \sum_{j \neq i} v_j(f^*(v)).$$

Theorem 1 *An equal sharing mechanism is dominant strategy incentive compatible and budget-balanced. Further, if at every type profile $v \equiv (v_1, \ldots, v_n) \in V$, we have $\sum_{i \in N} v_i(f^*(v)) \geq 0$, then the equal sharing mechanism is also individually rational.*

Proof: Let $(f^*, ((r_1^*, t_1^*), \ldots, (r_n^*, t_n^*)))$ be an equal sharing mechanism. Fix $i \in N$ and $v_{-i} \in V_{-i}$. For any $v_i, \in V_i$, agent i chooses a report v_i' to maximize

$$r_i^*(v_i', v_{-i}) v_i(f^*(v_i', v_{-i})) - t_i^*(v_i', v_{-i}) = \frac{1}{n} \left[v_i(f^*(v_i', v_{-i})) + \sum_{j \neq i} v_j(f^*(v_i', v_{-i})) \right],$$

which by the definition of f^* is maximized at $v_i' = v_i$. This implies dominant strategy incentive compatibility. Moreover, this payoff is clearly nonnegative whenever $\sum_{i\in N} v_i(f^*(v)) \geq 0$ which implies individual rationality. ■

Observe that, as the proof demonstrates, the equal sharing mechanism divides the welfare generated from the mechanism equally among the agents. Also note that it is simple to extend the above argument to accommodate costly alternatives. Here, there is a cost $\kappa(a)$ associated with each alternative a and the principal implements an scf f^κ which satisfies

$$f^\kappa(v) \in \operatorname*{argmax}_{a\in A} \left\{ \sum_{i\in N} v_i(a) - \kappa(a) \right\},$$

for each $v \equiv (v_1, \ldots, v_n) \in V$. By augmenting transfers of the equal sharing mechanism to include an equal share of the cost of the alternative $\kappa(f^\kappa(v))/n$, the contract will raise exactly the amount required to cover the cost.

4 Implementing Aggregate Utility Maximizers

Theorem 1 showed that using linear contracts allows for efficient redistribution. We now show that employing linear contracts significantly expands the set of implementable scfs over those that can be achieved using quasi-linear transfers.

To show this, we need to introduce some new notation. Given a type profile $v \equiv (v_1, \ldots, v_n) \in V$, we can define a vector $v^a \equiv (v_1(a), \ldots, v_n(a)) \in \mathbb{R}^n$ for each alternative $a \in A$. This is the utility vector of the agents if alternative a is chosen as the outcome. Given a type space V, it induces a set of permissible utility vectors for each alternative. We will denote the set of utility vectors v^a of alternative a as $\mathcal{U}^a$.

We will now define the notion of an aggregate utility function. Define the following set

$$\mathcal{X} := \{(a, x) : a \in A, x \in \mathcal{U}^a\}.$$

An **aggregate utility function** is a map $W : \mathcal{X} \to \mathbb{R}$. An aggregate utility function W is **monotone** if for every $a \in A$ and every $x, y \in \mathcal{U}^a$ such that $y \geq x$, we have $W(a, y) \geq W(a, x)$.[4]

Definition 5 *A social choice function f is an* **aggregate utility maximizer (AUM)** *if there exists a monotone aggregate utility function $W : \mathcal{X} \to \mathbb{R}$ such that at every profile $v \in V$, we have*

$$f(v) \in \operatorname*{argmax}_{a\in A} W(a, v^a).$$

[4]For any $x, y \in \mathbb{R}^n$, if $x_i \geq y_i$ for all $i \in N$, we write $x \geq y$.

Further, an AUM f satisfies consistent tie-breaking if there exists a linear ordering P on A such that at every profile $v \in V$, $f(v)$ is the maximum alternative in the set $\{a \in A : W(a, v^a) \geq W(b, v^b)\ \forall\ b \in A\}$ with respect to the linear order P.

This class of scfs include a number of commonly used social welfare functions. Below are a few examples.

1. **Affine Maximizers.** An scf f is an affine maximizer if there exist non-negative constants $(\gamma_1, \ldots, \gamma_n) \in \mathbb{R}^n_+ \setminus \{0\}$ and a map $\kappa : A \to \mathbb{R}$ such that for all $v \in V$
$$f(v) \in \operatorname*{argmax}_{a \in A} \left[\sum_{i \in N} \gamma_i v_i(a) - \kappa(a) \right].$$

2. **Generalized Utilitarianism.** f is a generalized utilitarian scf if there exist for every agent $i \in N$, an increasing function $g_i : \mathbb{R} \to \mathbb{R}$ such that for all $v \in V$
$$f(v) \in \operatorname*{argmax}_{a \in A} \left[\sum_{i \in N} g_i(v_i(a)) \right].$$
Note that an affine maximizer is a special case of a generalized utilitarian scf where g_i's are taken to be linear functions.

3. **Generalized Gini.** At any type profile $v \in V$ and any $a \in A$, order the utility numbers $(v_1(a), \ldots, v_n(a))$ and let $v^a_{(k)}$ denote the k-th lowest of these utility numbers. f is a generalized Gini scf if there exist $(\gamma_1, \ldots, \gamma_n) \in \mathbb{R}^n_+ \setminus \{0\}$ such that $\gamma_1 \geq \ldots \geq \gamma_n \geq 0$ and for all $v \in V$
$$f(v) \in \operatorname*{argmax}_{a \in A} \left[\sum_{i \in N} \gamma_i v^a_{(i)} \right].$$

4. **Max-min/Rawlsian.** f is a max-min scf if for all $v \in V$
$$f(v) \in \operatorname*{argmax}_{a \in A} \min_{i \in N} v_i(a).$$
Note that a max-min scfs is a special case of the generalized Gini scf, where we set $\gamma_1 = 1$ and $\gamma_i = 0$ for all $i \neq 1$.

The following result shows that an AUM with consistent tie-breaking is implementable if the type space is finite. Bikhchandani et al. (2006) showed (see their supplemental material) that Rawlsian scfs may not be implementable using a quasi-linear mechanism. Indeed, if the set of alternatives is finite and the type space is *unrestricted*, Roberts (1979) has shown that every scf that can be implemented using a quasi-linear mechanism must be an affine maximizer. Hence, these examples illustrate that the set of implementable scfs is significantly enlarged by considering linear contracts.

THEOREM **2** *Suppose the type space is finite. Then, every AUM with consistent tie-breaking is implementable.*

The proof of Theorem 2 is given in Appendix 1. It uses a general characterization of implementable scfs that we establish in Theorem 3 in the next section.

5 Characterizing Implementable SCFs

We now characterize the set of scfs that are implementable with linear contracts. The characterization yields an important property of linear contracts in finite type spaces: Every scf that can be implemented using a contingent contract can also be implemented using a linear contract. We show this result by providing an implicit characterization of implementable scfs.

We first derive an intuitive necessary condition for an scf to be implementable using a contingent contract. Given an scf f, for every $i \in N$ and for every $v_{-i} \in V_{-i}$, we define two binary relations $\succeq^f_{v_{-i}}$ and $\succ^f_{v_{-i}}$ on V_i as follows. For notational convenience, we write $\succeq^f \equiv \succeq^f_{v_{-i}}$ and $\succ^f \equiv \succ^f_{v_{-i}}$; the dependence on v_{-i} is implicitly implied. Fix an $i \in N$ and $v_{-i} \in V_{-i}$. For any, $v_i, v_i' \in V_i$, we define

$$v_i' \succeq^f v_i \quad \text{if} \quad v_i'(f(v_i, v_{-i})) \geq v_i(f(v_i, v_{-i})).$$

Further, for any $v_i, v_i' \in V_i$, we define $v_i' \succ^f v_i$ if $v_i'(f(v_i, v_{-i})) > v_i(f(v_i, v_{-i}))$.

A few comments about these binary relations are in order. In words, $v_i' \succeq^f v_i$ if the type v_i' gets a higher utility than type v_i from the alternative chosen by the scf f for the latter type. Clearly, the relation $\succeq^f$ is reflexive. However, note that the relation is neither antisymmetric, complete nor transitive. It is entirely possible that for types $v_i' \neq v_i$, $v_i'(f(v_i, v_{-i})) \geq v_i(f(v_i, v_{-i}))$ and $v_i(f(v_i', v_{-i})) \geq v_i'(f(v_i', v_{-i}))$ both hold simultaneously (even with either or both of the inequalities being strict) which implies that $\succeq^f$ need not be antisymmetric. Notice that this can happen because the utilities of the two types are being compared for two, potentially different alternatives. Similarly, it is easy to construct examples where $\succeq^f$ is not complete or transitive.

DEFINITION **6** *An scf f is* **acyclic** *if for every agent $i \in N$, for every v_{-i}, and for every sequence of types $v_i^1, \ldots, v_i^k \in V_i$ with $v_i^1 \succeq^f \ldots \succeq^f v_i^k$, we have $v_i^k \not\succ^f v_i^1$.*

In words, f is acyclic if there does not exist a cycle in the relation $\succeq^f$ where the relation is strict for at least one pair of types in the cycle. The simple lemma below shows that this condition is necessary for implementability by a contingent contract.

LEMMA **1** *If an scf is implementable by a contingent contract, it is acyclic.*

Proof: Let f be an scf that is implementable by contingent contracts $(s_1, \ldots, s_n)$. Fix $v_{-i} \in V_{-i}$ and consider a sequence of types $v_i^1, \ldots, v_i^k \in V_i$ for agent i, such that $v_i^1 \succeq^f \ldots \succeq^f v_i^k$. Hence, $v_i^j(f(v_i^{j+1}, v_{-i})) \geq v_i^{j+1}(f(v_i^{j+1}, v_{-i}))$ for all $j \in \{1, \ldots, k-1\}$. Pick any $j \in \{1, \ldots, k-1\}$. Since s_i implements f, incentive compatibility yields

$$s_i(v_i^j(f(v_i^j, v_{-i})), v_i^j, v_{-i}) \geq s_i(v_i^j(f(v_i^{j+1}, v_{-i})), v_i^{j+1}, v_{-i}). \tag{1}$$

Further, $v_i^j(f(v_i^{j+1}, v_{-i})) \geq v_i^{j+1}(f(v_i^{j+1}, v_{-i}))$ and monotonicity of s_i in the first argument imply that

$$s_i(v_i^j(f(v_i^{j+1}, v_{-i})), v_i^{j+1}, v_{-i}) \geq s_i(v_i^{j+1}(f(v_i^{j+1}, v_{-i})), v_i^{j+1}, v_{-i}). \tag{2}$$

Adding Inequalities (1) and (2) gives

$$s_i(v_i^j(f(v_i^j, v_{-i})), v_i^j, v_{-i}) \geq s_i(v_i^{j+1}(f(v_i^{j+1}), v_{-i}), v_i^{j+1}, v_{-i}).$$

Summing over $j \in \{1, \ldots, k-1\}$ and telescoping, we get

$$\begin{aligned} s_i(v_i^1(f(v_i^1, v_{-i})), v_i^1, v_{-i}) &\geq s_i(v_i^k(f(v_i^k, v_{-i})), v_i^k, v_{-i}) \\ &\geq s_i(v_i^k(f(v_i^1, v_{-i})), v_i^1, v_{-i}), \end{aligned}$$

where the last inequality follows from incentive compatibility of s_i. But monotonicity of s_i in the first argument implies that $v_i^1(f(v_i^1, v_{-i})) \geq v_i^k(f(v_i^1, v_{-i}))$. Hence, $v_i^k \not\succ v_i^1$, and this implies that f is acyclic. ■

The following theorem shows that for finite type space this condition is also sufficient for implementation by a contingent contract. Moreover, the theorem shows that acyclicity is sufficient for implementation by a linear contract. The proof is in Appendix 1.

THEOREM **3** *Suppose the type space is finite. Then, for any scf f, the following are equivalent.*

1. *f is implementable by a contingent contract.*
2. *f is acyclic.*
3. *f is implementable by a linear contract.*

Remark. For every acyclic scf, the proof explicitly constructs a linear contract that implements it. Under a mild condition on the type space, we show that a linear contract can be

constructed such that the resulting mechanism is individually rational and the transfer of each agent is non-negative (see the remark immediately following the proof). Further, the linear mechanisms that we construct have the property that the royalties rates r_i lie in $(0, 1]$. Hence, for implementation, the planner neither needs to make payments to nor take away large amount of utility from the agents. [5]

The finiteness of the type space is required for the equivalence between implementability by contingent and linear contracts. This can be seen from the following simple example of a single agent with a countably infinite type space where Theorem 3 breaks down.

EXAMPLE **1**

Consider a single agent with the following countably infinite type space

$$V_1 = \left\{v_1^2, v_1^3, \dots\right\} \cup \left\{v_1^\infty\right\}.$$

Suppose the set of alternatives A has equal cardinality and consider an scf f which satisfies

$$f(v_1^k) \neq f(v_1^{k'}) \qquad \text{for all } k \neq k'.$$

Define the type space such that

$$v_1^k(f(v_1^{k'})) = \begin{cases} \frac{2}{k'} & \text{if } k' < k, \\ \frac{1}{k} & \text{if } k' = k, \\ \frac{1}{2k'} & \text{if } k' > k, \\ 0 & \text{if } k' = \infty. \end{cases}$$

Finally, we define utility for type v_1^∞ as

$$v_1^\infty(f(v_1^{k'})) = \begin{cases} \frac{2}{k'} & \text{if } k' < \infty, \\ 1 & \text{otherwise.} \end{cases}$$

It is easy to see that f is acyclic. This is because $v_1^k \succ v_1^{k'}$ for all $k' < k \leq \infty$ as

$$v_1^k(f(v_1^{k'})) = \frac{2}{k'} > \frac{1}{k'} = v_1^{k'}(f(v_1^{k'})).$$

Moreover, when $k < k' < \infty$ then $v_1^k \not\succeq v_1^{k'}$ as

$$v_1^k(f(v_1^{k'})) = \frac{1}{2k'} < \frac{1}{k'} = v_1^{k'}(f(v_1^{k'})).$$

[5] This is true even for standard quasi-linear mechanisms - every implementable scf can be implemented (under reasonable conditions) using individually rational and non-negative transfers (Kos and Messner, 2013).

Finally, $v_1^k \not\succeq v_1^\infty$

$$v_1^k(f(v_1^\infty)) = 0 < 1 = v_1^\infty(f(v_1^\infty)).$$

Lemma 3 in the appendix shows that acyclicity remains a sufficient condition for implementability if the type space is countable and, hence, f can be implemented using a contingent contract.

We now show that f cannot be implemented by a linear contract. Let us assume to the contrary that it is implementable by (r_1, t_1). Then adding the two incentive compatibility conditions for types v_1^k, v_1^∞ misreporting as each other, we get

$$\begin{aligned} & r_1(v_1^k)[v_1^k(f(v_1^k)) - v_1^\infty(f(v_1^k))] + r_1(v_1^\infty)[v_1^\infty(f(v_1^\infty)) - v_1^k(f(v_1^\infty))] \geq 0 \\ \Rightarrow & \frac{r_1(v_1^\infty)}{r_1(v_1^k)} \geq -\frac{\frac{1}{k} - \frac{2}{k}}{1-0} = \frac{1}{k}. \end{aligned}$$

Similarly, incentive compatibility for types v_1^k, v_1^{k+1} implies

$$\begin{aligned} & r_1(v_1^k)[v_1^k(f(v_1^k)) - v_1^{k+1}(f(v_1^k))] + r_1(v_1^{k+1})[v_1^{k+1}(f(v_1^{k+1})) - v_1^k(f(v_1^{k+1}))] \geq 0 \\ \Rightarrow & \frac{r_1(v_1^{k+1})}{r_1(v_1^k)} \geq -\frac{\frac{1}{k} - \frac{2}{k}}{\frac{1}{k+1} - \frac{1}{2(k+1)}} = \frac{2(k+1)}{k}. \end{aligned}$$

Multiplying inequalities for succeeding $k = 2, \ldots, K-1$, we get

$$\frac{r_1(v_1^K)}{r_1(v_1^2)} \geq 2^{K-3}K.$$

Combining inequalities, we get

$$r_1(v_1^\infty) \geq \frac{r_1(v_1^K)}{K} \geq 2^{K-3} r_1(v_1^2).$$

Taking the limit $K \to \infty$, we observe that the right side diverges, which implies that $r_1(v_1^\infty)$ must be ∞ which is a contradiction. Hence, f cannot be implemented using a linear contract.

We now contrast our acyclicity condition with the seminal characterization of implementation with quasi-linear transfers due to Rochet (1987). His theorem is presented below.

THEOREM 4 (**Rochet (1987)**) *A scf f is implementable by quasi-linear transfers if and only if for every agent $i \in N$, for every $v_{-i} \in V_{-i}$, and for every finite sequence of types $v_i^1, \ldots, v_i^k \in V_i$ with $v_i^{k+1} = v_i^1$, the following **cycle monotonicity** condition holds*

$$\sum_{j=1}^{k} \left[v^{j+1}(f(v_i^{j+1}, v_{-i})) - v_i^{j+1}(f(v_i^j, v_{-i}))\right] = \sum_{j=1}^{k} \left[v^j(f(v_i^j, v_{-i})) - v_i^{j+1}(f(v_i^j, v_{-i}))\right] \geq 0.$$

In contrast to our notion of acyclicity over types, Rochet (1987) provided the following acyclicity condition over alternatives as a necessary condition for implementation with quasi-linear transfers. Fix a given agent i and a profile of reports $v_{-i} \in V_{-i}$. For a finite sequence of types $v_i^1, \ldots, v_i^k \in V_i$, define a binary relation $\succ_a^f$ over alternatives $\{a_i^j = f(v_i^j, v_{-i})\}_{j=1}^k$, as follows

$$a_i^{j'} \succ_a^f a_i^j \text{ whenever } v_i^j(a_i^{j'}) > v_i^j(a_i^j).$$

A necessary condition for f to be implementable by quasi-linear transfers is that $\succ_a^f$ is acyclic for all finite sequences of types for all agents $i \in N$ and all profiles of reports $v_{-i} \in V_{-i}$. The necessity of this condition can easily be seen by examining the cycle monotonicity condition in Theorem 4. If there was a cycle in $\succ_a^f$ for a sequence of types then each term in the first summation of the cycle monotonicity condition for that sequence would be negative. This condition is however not sufficient for implementability by quasilinear transfers and is neither necessary nor sufficient for implementation by contingent contracts.

By contrast, our acyclicity condition is defined on the set of types and clearly must be necessary for quasi-linear implementation. This necessity can be seen by observing that if there is a cycle then each term in the second summation of the cycle monotonicity condition would be nonpositive with at least one being negative.

Given the equivalence in terms of implementability between contingent and linear mechanisms, a natural question to ask is whether the payoffs from every contingent mechanism can also be achieved by a linear mechanism. More precisely, given an scf f and contingent contracts $(s_1, \ldots, s_n)$ that implement it, we ask if there exist linear contracts $((r_1, t_1), \ldots, (r_n, t_n))$ that implement f such that

$$s_i(v_i(f(v_i, v_{-i})), v_i, v_{-i}) = r_i(v_i, v_{-i})v_i(f(v_i, v_{-i})) - t_i(v_i, v_{-i}) \text{ for all } i \in N, v_i \in V_i, v_{-i} \in V_{-i}.$$

Note that this requirement is only for payoffs on the equilibrium path. The following single agent example shows that this payoff equivalence does not hold.

EXAMPLE **2**

Consider a single agent with type space $V_1 := \{v_1^1, v_1^2, v_1^3\}$. Let the set of alternatives be $A := \{a^1, a^2, a^3\}$. The utilities of each type from the different alternatives is shown in Table 1 below.

Consider the scf f defined as: $f(v_1^j) = a^j$ for $j \in \{1, 2, 3\}$. This can be implemented by

	v_1^1	v_1^2	v_1^3
a^1	30	20	10
a^2	30	20	10
a^3	20	10	10

Table 1: Type space for which revenue equivalence fails.

a contingent contract s_1, which is as follows:

$$\begin{aligned}
&s_1(30, v_1^1) = 20, s_1(20, v_1^1) = 5, s_1(10, v_1^1) = 1,\\
&s_1(30, v_1^2) = 16, s_1(20, v_1^2) = 15, s_1(10, v_1^2) = 1,\\
&s_1(20, v_1^3) = 10, s_1(10, v_1^3) = 5.
\end{aligned}$$

Now, suppose there is a payoff equivalent linear contract (r_1, t_1) that implements f. Then,

$$r_1(v_1)v_1(f(v_1)) - t_1(v_1) = s_1(v_1(f(v_1)), v_1) \text{ for all } v_1 \in V_1.$$

Incentive compatibility of the linear mechanism would then imply that for all v_1, v_1',

$$s_1(v_1'(f(v_1')), v_1') - s_1(v_1(f(v_1)), v_1) \leq r_1(v_1')[v_1'(f(v_1')) - v_1(f(v_1'))]. \tag{3}$$

Taking $v_1' = v_1^2$ and $v_1 = v_1^3$ in Inequality (3), we get $r_1(v_1^2) \geq 1$. Taking $v_1' = v_1^2$ and $v_1 = v_1^1$ in Inequality (3), we get $r_1(v_1^2) \leq \frac{1}{2}$, which gives us a contradiction.

The usual revenue/payoff equivalence in quasi-linear environments (Krishna, 2009) requires that two quasi-linear transfers implementing the same scf must differ in payoffs by a constant. It is well known that in finite type spaces, this does not hold. However, the payoff equivalence that we seek is across two *classes* of contracts implementing the same scf. The failure of payoff equivalence in the above example is not driven by finite type space restriction and it is easy to construct examples with a continuum of types.

5.1 An Explicit Characterization

If the type space is finite and rich, we can provide a complete description of the set of scfs implementable using linear contracts. The richness of type space that we require is the following.

DEFINITION **7** *The type space V is* **rich** *if the set of profiles of utility vectors is $\mathcal{U}^a \times \mathcal{U}^b \times \ldots$*

This richness condition requires that every combination of utility vectors is a feasible type profile. For instance, if v^a and v'^a are two utility vectors corresponding to alternative a in $\mathcal{U}^a$

and (v^a, v^{-a}) is a profile of utility vectors at a type profile, then the profile of utility vectors v'^a, v^{-a} must correspond to a valid type profile in the type space V. Let $\mathcal{U} := \mathcal{U}^a \times \mathcal{U}^b \times \ldots$.

We now introduce a new condition on the scfs.

DEFINITION **8** *An scf f satisfies* **binary independence** *if for every distinct pair of alternatives $a, b \in A$ and every $v, v' \in V$ such that $v^a = v'^a, v^b = v'^b$, $f(v) = a$ implies that $f(v') \neq b$.*

Binary independence requires that if a is chosen over b as the outcome by an scf at a type profile, then b cannot be chosen at a different type profile in which the utility vectors corresponding to a and b are not changed. In other words, the scf must evaluate a and b at any type profile independent of utility vectors of other alternatives. Essentially, this is a consistent tie-breaking condition for arbitrary scfs (which need not be aggregate utility maximizers). It is in the spirit of binary independence used in the social choice theory literature (d'Aspremont and Gevers, 2002) from where we borrow the terminology.

We now show that under richness and finiteness of type space, we can strengthen Theorem 3 significantly using the following weakening of acyclicity.

DEFINITION **9** *An scf f is* **2-acyclic** *if for every agent $i \in N$, for every $v_{-i} \in V_{-i}$, and for every pair of types $v_i, v_i' \in V_i$ with $v_i \succeq^f v_i'$, we have $v_i' \not\succ^f v_i$.*

THEOREM **5** *Suppose the type space is rich and finite. Then, the following are equivalent.*

1. *f is an aggregate utility maximizer with consistent tie-breaking.*
2. *f is implementable and satisfies binary independence.*
3. *f is 2-acyclic and satisfies binary independence.*

Theorem 5 shows that AUMs with consistent tie-breaking are the only implementable scfs satisfying binary independence under the additional richness condition. Further, it shows that 2-acyclicity, a much weaker condition than acyclicity, is equivalent to implementability under binary independence.

It is interesting to compare this characterization to similar characterizations in the standard quasi-linear environments. Roberts (1979) showed that affine maximizers are the only implementable scfs in such environments under some conditions on the type space and set of alternatives. Though Theorem 5 can be viewed as counterpart of that result in the contingent contract environment, there are significant differences. We require type space to be finite and rich while Roberts (1979) required the type space to be whole of $\mathbb{R}^{|A|}$, where A is a finite set of alternatives. Roberts (1979) required at least three alternatives and the scf to be onto. Though we do not require this, we need binary independence.

6 The General Model with Uncertainty

In this section, we present the general model with uncertainty and discuss how the results in the previous sections extend to this environment. For this, we will need some additional notation. The type v_i of the agent now determines the distribution of the ex-post utility that an agent receives from an alternative a. We denote by u_i, the random variable for agent i corresponding to the ex-post utility. At the interim stage (that is, after realization of the type and before an alternative is chosen), this utility is not known to the agent and the mechanism designer. It is assumed that when agent i has type v_i, his ex-post utility u_i from alternative a is drawn from $\mathbb{R}$ with cumulative distribution $G^a_{v_i}$ which depends both on the true type and the alternative. Note that since the utility is a random variable, its realization need not reveal the type of the agent.[6] In a minor abuse of notation, we use $v_i(a)$ to denote the expected utility from alternative a or $v_i(a) = \int_{\mathbb{R}} u_i dG^a_{v_i}(u_i)$.

We will impose the following restriction on the distribution of utilities.

Definition **10** *The distributions of utilities are ordered by first order stochastic dominance or simply* **ordered** *if for all i, $v_i, v_i' \in V_i$ and for all $a \in A$, we have*

$$\text{either } G^a_{v_i} \succeq_{FOSD} G^a_{v_i'} \text{ or } G^a_{v_i'} \succeq_{FOSD} G^a_{v_i},$$

where $\succeq_{FOSD}$ is the first-order stochastic dominance relation.

The above ordering requirement says that for every agent i and every alternative $a \in A$, the types in V_i can be ex-ante ordered using the $\succeq_{FOSD}$ relation. Note that this does not imply that the ordering of types has to be the same across the different alternatives. To the best of our knowledge, most of the theoretical work on mechanism design with contingent contracts requires this assumption.[7] Importantly, the deterministic environment (which corresponds to the distributions being degenerate) we have studied in the previous sections is ordered in the above sense.

Finally, as in the deterministic case, we assume that there are no duplicate types. In other words, for all agents $i \in N$, there are no two types $v_i, v_i' \in V_i$ such that $v_i(a) = v_i'(a)$ for all $a \in A$.[8]

[6] Of course, if the principal knew the prior distribution over the agents' types, the realized utility would allow him to update the prior. By contrast, if the principal does not know the type distribution, he will not be able to make inference (dominant strategy implementation is appropriate for these cases). That said, we allow the supports of the distributions of utilities to vary over different alternatives. Hence, even without prior knowledge of how the types are distributed, there may be certain realizations of utility from which the principal can back out the type of the agent.

[7] Often, the stronger assumption of affiliation (Milgrom and Weber, 1982) is made instead (DeMarzo et al., 2005; Gorbenko and Malenko, 2010).

[8] Again, this assumption is made for expositional purposes and is not required for the results.

We can now define an scf and the contracts analogously to the deterministic environment. As before, an scf is a mapping $f : V \to A$. Linear contracts are defined identically and consist of functions $r_i : V \to (0, \infty)$ and $t_i : V \to \mathbb{R}$ for each agent i. Similarly, contingent contracts are mappings $s_i : \mathbb{R} \times V \to \mathbb{R}$ for each agent i which is strictly increasing in the first argument.

Our notion of (dominant strategy) implementation can be adapted in the natural way. Agents now compute their *expected utility* before reporting their types. The definition of implementability by linear contracrs looks identical to the deterministic case with the only difference being that the $v_i(\cdot)$ in the incentive compatibility constraints now denotes the expected utility. Implementation by contingent contracts is defined below.

DEFINITION **11** *An scf f is* **implementable by a contingent contract** *if there exist contingent contracts $(s_1, \ldots, s_n)$ such that*

$$\int_{\mathbb{R}} s_i(u_i, v_i, v_{-i}) dG_{v_i}^{f(v_i, v_{-i})}(u_i) \geq \int_{\mathbb{R}} s_i(u_i, v_i', v_{-i}) dG_{v_i}^{f(v_i', v_{-i})}(u_i),$$

$$\forall\ i,\ v_i, v_i' \in V_i \text{ and } v_{-i} \in V_{-i}.$$

In this case, we say that the contingent contracts $(s_1, \ldots, s_n)$ implement f and the contingent mechanism $(f, s_1, \ldots, s_n)$ is incentive compatible.

Since the alternative is chosen by the principle before the utility is realized, the efficient scf can be taken to be the one that maximizes the sum of the expected utilities of the agents. It is easy to see that Theorem 1 generalizes as the equal sharing mechanism will continue to achieve efficiency with budget balance. Note that for this result, there is no need for the ordering condition on the distributions.

For the remaining results, however, we need the distributions to be ordered. This is because acyclicity only characterizes implementability under this condition. Note that, the definition of acyclicity remains unchanged with, once again, the difference being that the $v_i(\cdot)$'s used to define the relations $\succeq^f$ and $\succ^f$ are expected utilities. Note also that since the type space is assumed to be ordered $v_i(a) \geq (>) v_i'(a)$ is equivalent to $G_{v_i}^a \succeq_{FOSD} (\succ_{FOSD}) G_{v_i'}^a$. Finally, observe that if the type space is ordered, acyclicity remains necessary for implementation. This is because for ordered types $v_i', v_i \in V_i$, the following holds

$$v_i' \succeq^f v_i \quad \Longrightarrow \quad \int_{\mathbb{R}} s_i(u_i, v_i, v_{-i}) dG_{v_i'}^{f(v_i, v_{-i})}(u_i) \geq \int_{\mathbb{R}} s_i(u_i, v_i, v_{-i}) dG_{v_i}^{f(v_i, v_{-i})}(u_i),$$

where the inequality follows from the monotonicity of s_i in u_i and the fact that $G_{v_i'}^{f(v_i, v_{-i})}$ first order stochastically dominates $G_{v_i}^{f(v_i, v_{-i})}$. As in the proof of Lemma 1, this combined with incentive compatibility ensures that every implementable scf must be acyclic.

As with the case of efficiency, a natural way to define aggregate utility maximizers is in terms of expected utilities and, with this definition, Theorem 2 continues to hold as stated. The definition of utility vector of the agents corresponding to a given alternative a, $\mathcal{U}^a$, will remain the same with the $v_i(\cdot)$'s now being expected utilities.

The characterization results, Theorem 3 and Theorem 5, hold verbatim with the adjusted definitions of acyclicality, richness and binary independence. Finally, uncertainty can be introduced into the Examples 1 and 2.

7 Related Literature

Mechanism design with contingent contracts originated with the literature on security auctions (Hansen, 1985; Riley, 1988). This paper has been partly inspired by the recent work which discuss the revenue ranking of auctions conducted with different contingent contracts (DeMarzo et al., 2005; Che and Kim, 2010; Abhishek et al., 2012). These papers study how a seller's revenue is affected by the "steepness" of securities that are admissible as bids. In contrast to the work on security auctions, our focus is on a general mechanism design environment and our goal is to characterize dominant strategy incentive compatibility. Additionally, since we do not focus on auctions, we do not need the space of admissible contingent contracts to be ranked - securities are completely ordered and better securities provide a higher expected payoff to the seller irrespective of bidder type. This restriction is required in security auctions to ensure that a winner can be declared based on the bids but before the utility is realized. In other words, the linear contracts we consider (which cannot not be ranked ex-ante) are explicitly prohibited in the security auctions literature. For a recent survey of work on auctions with contingent payments, see Skrzypacz (2013).

This paper is related to the literature on implementation with quasi-linear transfers which originated with Rochet (1987) whose characterization we have discussed earlier. Recent contributions (Bikhchandani et al., 2006; Saks and Yu, 2005; Ashlagi et al., 2010; Mishra and Roy, 2013) to this literature investigate conditions that are weaker than cycle monotonicity which characterize implementability in such environments with finite alternatives. Our characterization result Theorem 5 using 2-acyclicity (for rich type spaces) is similar in spirit to these characterizations. This result is also related to Roberts (1979), who offers the counterpart of such a characterization with quasi-linear mechanisms (see also Mishra and Sen (2012); Carbajal et al. (2013)).

The Myerson-Satterthwaite theorem (Myerson and Satterthwaite, 1983) has started a literature that investigates various special conditions under which budget-balance, efficiency, incentive compatibility, and individual rationality is compatible. For instance, Cramton et al. (1987) show that it is possible while dissolving a partnership, where different agents have

predefined property rights over a resource. Similarly, Suijs (1996) shows that this is possible in queueing models and Mitra and Sen (2010) show that this is possible in some specific multi-unit auction problems.[9] In contrast to these results for specific settings, Theorem 1 shows that these goals are achievable in general with linear contracts.

Like this paper, Rahman (2011) characterizes implementation in an environment where the principal can observe and condition the mechanism on a noisy signal which is correlated with the agent's type. The environment he considers differs fundamentally from ours in at least two important respects. Firstly, in his model, both the scf and the payments are functions of the signal and the agent's report. Hence, the signals in his model depend only on the agent's type and not on the allocation. By contrast, in our setting, the scf depends only on the reports whereas the contracts depend additionally on the realized utility. Secondly, while we consider general securities, he restricts attention to quasilinear transfers. That said, it should be noted that he considers a signal structure which is more general than ours as he does not impose the ordering condition. A challenging and fruitful problem for future research would be to characterize implementation by contingent contracts in an environment where utility distributions depend on allocations but are otherwise unrestricted as in Rahman (2011).

The task scheduling problem in algorithmic mechanism design is related to the implementation of AUMs (Theorem 2). Here, a principal is trying to minimize the total time taken to complete a set of tasks by allocating them to a set of agents whose private information is the time they take to complete the different tasks. Nisan and Ronen (2001) consider a linear environment and argue that no quasilinear transfers can achieve the optimal time. Instead they show that the optimum can be achieved if the principal can condition payments on the realized times of completion. Our characterization could potentially be useful in extending the results of Nisan and Ronen (2001) to general nonlinear environments. Here, the agents may have synergies in production– groups of tasks may be completed in less than the sum of time they would take to complete each task in the group individually. Additionally, the principal may have more complicated preferences in which certain tasks take precedence over others. We leave this interesting problem for future research.

[9] A recent literature in computer science, while still restricting to the standard quasi-linear mechanisms, tries to achieve the "second-best" (as much budget-balance as possible while maintaining efficiency, incentive compatibility, and individual rationality) by taking the worst-case approach - contributions to this end are Moulin (2009) and Guo and Conitzer (2009).

8 Concluding Remarks

In this paper, we study a general version of the classic dominant strategy implementation problem introduced by Rochet (1987). We consider environments where the principal can offer contracts that depend on the (random) realized utility of the agents, the distribution of which is a function of the private type of the agent and the outcome. Our model nests the standard deterministic quasilinear setting. We focus on linear contracts which we motivate by providing a number of applications. We then provide results which characterize the set of scfs implementable by linear contracts and, additionally, provide a foundation for restricting attention to these contracts. Following Rochet (1987), there has been a large and insightful body of work in dominant strategy quasilinear implementation. We hope that this paper spurs some interest in studying implementation with contingent contracts. To this end, we conclude with some discussion about our results and few suggestions for future research.

The proof of our characterization result Theorem 3 uncovers a parallel with Afriat's theorem of revealed preference in consumer theory.[10] The acyclicity condition we use to characterize implementability is analogous to the Generalized Axiom of Revealed Preference (Varian, 1982) which is necessary and sufficient condition for a finite price consumption data set to be rationalized by a utility maximizing consumer. Additionally, Afriat's theorem (Afriat, 1967; Varian, 1982) shows that a data set can be rationalized by a utility function if and only if it can be rationalized by a concave utility function. Analogously, we show that acyclicity is necessary and sufficient for implementability using either contingent or linear contingent contracts. By contrast, implementability by quasilinear transfers is characterized by cycle monotonicity which is a stronger condition than acyclicity.[11]

An assumption in our model is that all of the realized utility of the agents is contractible. While this is appropriate in many settings, it is a strong assumption for others. Theorem 6 in Appendix 2, extends Theorem 3 to an environment where the realized utility is in two parts – contractible and noncontractible. We show that as long as both are comonotone, the result will continue to hold. Of course, an important extension for future work is to examine environments in which these are not comonotone where Theorem 3 does not hold in general.

Theoerem 7 in Appendix 2 shows that the equivalence of implementation between linear and contingent contracts holds in uncountable type spaces under additional smoothness conditions. However, the smoothness we require for this result is often absent in many practical applications such as auctions. We hope to conduct a more formal analysis of uncountable type spaces in the future. Here, a natural question is: When the equivalence of

[10] Beginning with (Rochet, 1987), there have been informal analogies made between these two problems.

[11] Implementability of an scf by quasilinear transfers can be considered to be analogous to rationalizability of choice data by quasilinear utility functions (Brown and Calsamiglia, 2007).

linear contracts and contingent contracts in Theorem 3 fails, is there is a different class of simple (nonlinear) contracts which are sufficient for implementation?

While we constructed a budget balanced and individually rational efficient linear mechanism, a natural direction of future research will be to identify a larger class of scfs that can be implemented with budget balanced linear contracts. In particular, an important class of scfs to focus on would be the class of affine maximizers.

Although we demonstrated that acyclicity is easy to check in the context of aggregate utility maximizers, it may, in principle, be difficult to verify for certain applications. This is because it requires checking for the absence of cycles of all finite lengths. Theorem 5 helped in this regard by showing that the substantially weaker condition 2-acyclicity is sufficient but only as long as the type space is rich. In Appendix 2, we show that 2-acyclicity is sufficient in certain commonly utilized settings even when the type space is not rich – linear one dimensional environments with uncountable types (Theorem 8) and linear two dimensional environments with countable types (Theorem 9).

Another interesting generalization would be to consider interdependent value settings. Even the efficient outcome is difficult to implement in this setting using quasi-linear mechanisms - Maskin (1992) shows that if the utility function of each agent satisfies a single crossing condition then the utilitarian efficient outcome can be implemented. However, Mezzetti (2004) has shown that using two-stage mechanisms that depend on the realized utilities of agents, the utilitarian efficient outcome can always be implemented even in the interdependent values model. We are not aware of work analyzing the implementability of Rawlsian scfs in an interdependent value setting.

Perhaps one of the reasons that contingent contracts have received limited attention is because of an observation of Crémer (1987). This observation states that by offering a very low share of the ex-post utility to the agents ($r_i(\cdot)$ close to zero), the principal can make the information rents negligible. In other words, by almost completely buying the agents, the principal can always get arbitrarily close to the first best. Of course, while this is a sound theoretical argument, it is seldom observed in real world for a number of reasons. For instance, the principal may be liquidity constrained and hence, may be unable to finance the necessary upfront payment to buy the agent. In other cases, as DeMarzo et al. (2005) argue, the agents may have to make noncontractible, fixed, costly investments in order for profits to be realized. If the ex-post payoffs offered by the contingent contract are too low, the agent may choose to just accept the upfront payment and not to undertake the investment. In practice, environments which feature contingent contracts often have such legal or practical restrictions on the set of contracts that the principle can offer. For these applications, our characterization of incentive compatibility is an important first step which can help in the derivation of optimal contracts.

More generally, contingent contracts are necessary to provide incentives in environments which feature both adverse selection and moral hazard- ex-post utilities are affected by types, alternatives *and actions*. A particular example of such an environment is Laffont and Tirole (1986) where a principal is trying to regulate the cost of an agent who has a private efficiency parameter and can reduce his cost by conducting costly unobservable effort. In their setting, the principal is permitted to use very general contracts and they surprisingly show that the second best solution can be achieved using simple linear contracts. An important but challenging issue for future research is to provide conditions characterizing implementable outcomes in the general environment considered in this paper but with additional moral hazard.

Appendix 1: Omitted Proofs and Examples

8.1 Proof of Theorem 3

Throughout the proof, we fix an agent i and type profile of other agents at v_{-i}. For notational convenience, we suppress the v_{-i} from notations everywhere. We begin the proof by noting that a consequence of acyclicity is that the type space can be partitioned. A type space V_i can be f-order-partitioned if there exists a partition $(V_i^1, \ldots, V_i^K)$ of the type space V_i such that

P1 for each $j \in \{1, \ldots, K\}$ and for each $v_i, v_i' \in V_i^j$, we have $v_i \not\succ^f v_i'$,

P2 for each $j \in \{1, \ldots, K-1\}$, for each $v_i \in V_i^j$, and for each $v_i' \in (V_i^{j+1} \cup \ldots \cup V_i^K)$, we have $v_i' \not\succeq^f v_i$.

We first show that any acyclic SCF f induces an f-ordered-partition of the type space.

Lemma **2** *Suppose the type space is finite and f is an acyclic SCF. Then, the type space can be f-ordered-partitioned.*

Proof: Let f be an acyclic scf. Consider any non-empty subset $V_i' \subseteq V_i$. A type v_i is maximal in V_i' with respect to $\succ^f$ if there exists no type $v_i' \in V_i'$ such that $v_i' \succ^f v_i$. Denote the set of types that are maximal in V_i' with respect to $\succ^f$ as $\widetilde{V_i'}$. Since f is acyclic, $\succ^f$ is acyclic. Since V_i' is finite, we conclude that $\widetilde{V_i'}$ is non-empty (Sen, 1970). Define

$$M(V_i') := \{v_i \in \widetilde{V_i'} : \ v_i' \not\succeq^f v_i \ \forall \ v_i' \in V_i' \setminus \widetilde{V_i'}\}.$$

We claim that $M(V_i')$ is non-empty. Assume for contradiction that $M(V_i')$ is empty. Choose $v_i^1 \in \widetilde{V_i'}$. Since $M(V_i')$ is empty, there exists $\bar{v_i}^1 \in V_i' \setminus \widetilde{V_i'}$ such that $\bar{v_i}^1 \succeq^f v_i^1$. Since $\bar{v_i}^1 \in V_i' \setminus \widetilde{V_i'}$, there exist a sequence of types $(v_i^2, \ldots, v_i^k)$ such that $v_i^2 \succ^f \ldots \succ^f v_i^k \succ^f \bar{v_i}^1 \succeq^f v_i^1$ and $v_i^2 \in \widetilde{V_i'}$. Since $v_i^2 \in \widetilde{V_i'}$ and $M(V_i')$ is empty, there must exist $\bar{v_i}^2 \in V_i' \setminus \widetilde{V_i'}$ such that $\bar{v_i}^2 \succeq^f v_i^2$. This process can be repeated. Since V_i' is finite, we will get a cycle of types satisfying $v_i \ldots \succeq^f \ldots \succ^f \ldots v_i$. Since f is acyclic, $v_i \not\succeq^f v_i$. But this contradicts the fact that $\succeq^f$ is reflexive. Hence, $M(V_i')$ is non-empty.

We note that for any $v_i, v_i' \in M(V_i')$, we have $v_i \not\succ^f v_i'$. Now, we recursively define the f-ordered partition of V_i. First, we set $V_i^1 := M(V_i)$. Having defined $V_i^1, \ldots, V_i^k$, we define $R^k := V_i \setminus (V_i^1 \cup \ldots \cup V_i^k)$. If $R^k \neq \emptyset$, then define $V_i^{k+1} := M(R^k)$ and repeat. If $R^k = \emptyset$, then $V_i^1, \ldots, V_i^k$ is an f-ordered partition of V_i by construction. ■

A consequence of Lemma 2 is that f satisfies the following property.

DEFINITION **12** *An scf f satisfies* **multiplier K-cycle monotonicity**, *where $K \geq 2$ is a positive integer, if there exists $\lambda_i : V_i \to (0, \infty)$ such that for all sequence of types $(v_i^1, \ldots, v_i^k)$ with $k \leq K$, we have*

$$\sum_{j=1}^{k} \lambda_i(v_i^j)\big[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j))\big] \geq 0, \tag{4}$$

where $v_i^{k+1} \equiv v_i^1$. An scf f is multiplier cycle monotone if it satisfies multiplier K-cycle monotonicity for all integers $K \geq 2$. In this case, we say λ_i makes f multiplier cycle monotone.

To show that f is multiplier cycle monotone, we construct a λ_i that makes it multiplier cycle monotone.

Constructing λ_i

We use Lemma 2 to construct the λ_i map recursively. Let f be an acyclic SCF and $(V_i^1, \ldots, V_i^K)$ be the f-ordered-partition according to Lemma 2. First, we set

$$\lambda_i(v_i) = 1 \ \forall \ v_i \in V_i^K. \tag{5}$$

Having defined $\lambda_i(v_i)$ for all $v_i \in (V_i^{k+1} \cup V_i^{k+2} \cup \ldots \cup V_i^K)$, we define $\lambda_i(v_i)$ for all $v_i \in V_i^k$.

Let C be any cycle of types $(v_i^1, \ldots, v_i^q, v_i^1)$ involving types in $(V_i^k \cup V_i^{k+1} \cup \ldots V_i^K)$ with at least one type in V_i^k and at least one type in $(V_i^{k+1} \cup \ldots \cup V_i^K)$. Let $\mathcal{C}$ be the set of all such cycles. For each cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1) \in \mathcal{C}$, [12] define

$$L(C) = \sum_{v_i^j \in C \cap (V_i^{k+1} \cup \ldots \cup V_i^K)} \lambda_i(v_i^j)\big[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j))\big] \tag{6}$$

$$\ell(C) = \sum_{v_i^j \in C \cap V_i^k} \big[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j))\big]. \tag{7}$$

We now consider two cases.

CASE 1. If $L(C) \geq 0$ for all $C \in \mathcal{C}$, then we set $\lambda_i(v_i) = 1$ for all $v_i \in V_i^k$.

CASE 2. If $L(C) < 0$ for some $C \in \mathcal{C}$, we proceed as follows. Since V_i is f-ordered partitioned, for every $v_i \in V_i^k$ and $v_i' \in (V_i^{k+1} \cup \ldots \cup V_i^K)$, we have $v_i' \not\succeq^f v_i$ (Property P1 of f-ordered partition), and hence,

$$v_i(f(v_i)) - v_i'(f(v_i)) > 0.$$

[12] We will abuse notation to denote the set of types in a cycle C by C also.

Similarly, for every $v_i, v_i' \in V_i^k$, we have $v_i' \not\succ^f v_i$ (Property P2 of f-ordered partition), and hence,

$$v_i(f(v_i)) - v_i'(f(v_i)) \geq 0.$$

Then, for every $C \in \mathcal{C}$, we must have $\ell(C) > 0$ since C involves at least one type from V_i^k and at least one type from $(V_i^{k+1} \cup \ldots \cup V_i^k)$. Now, for every $v_i \in V_i^k$, define

$$\lambda_i(v_i) := \max_{C \in \mathcal{C}: L(C) < 0} \frac{-L(C)}{\ell(C)}. \tag{8}$$

We can thus recursively define the λ_i map.

PROPOSITION 1 *Suppose V_i is finite. If an scf f is acyclic, then λ_i makes f multiplier cycle monotone.*

Proof: Suppose f is acyclic. By Lemma 2, V_i can be f-ordered-partitioned. Let the induced partition of V_i be $(V_i^1, \ldots, V_i^K)$ and let λ be defined recursively as before using Equations (5) and (8). Consider any cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1)$. We will show that

$$\sum_{v_i^j \in C} \lambda_i(v_i^j) \left[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j)) \right] \geq 0. \tag{9}$$

If $C \subseteq V_i^K$, then $v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j)) \geq 0$ (by Property P1 above) and $\lambda_i(v_i^j) = \lambda_i(v_i^{j+1})$ for all $v_i^j, v_i^{j+1} \in C$. Hence, Inequality (9) holds.

Now, suppose Inequality (9) is true for all cycles $C \subseteq (V_i^{k+1} \cup \ldots V_i^K)$. Consider a cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1)$ involving types in $(V_i^k \cup \ldots \cup V_i^K)$. If each type in C is in V_i^k, then again $v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j)) \geq 0$ (by Property P1 above) and $\lambda_i(v_i^j) = \lambda_i(v_i^{j+1})$ for all $v_i^j, v_i^{j+1} \in C$. Hence, Inequality (9) holds. By our hypothesis, if all types in C belong to $(V_i^{k+1} \cup \ldots \cup V_i^K)$, then again Inequality (9) holds. So, assume that C is a cycle that involves at least one type from V_i^k and at least one type from $(V_i^{k+1} \cup \ldots \cup V_i^K)$. Let $\lambda_i(v_i) = \mu$ for all $v_i \in V_i^k$. By definition,

$$\sum_{v_i^j \in C} \lambda_i(v_i^j) \left[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j)) \right] = L(C) + \mu \ell(C) \geq 0,$$

where the last inequality followed from the definition of μ (Equation (8)). Hence, Inequality (9) again holds. Proceeding like this inductively, we complete the proof. ■

Using λ_i, we can define our linear contract that implements f. For this, we need to now define the transfers.

Constructing t_i

If λ_i makes f multiplier cycle monotone, then λ_i satisfies Inequality ((4)) for any cycle of types. Hence, by Rochet-Rockafellar cycle monotonicity characterization (Rochet, 1987; Rockafellar, 1970), there exists a map $W_i : V_i \to \mathbb{R}$ such that

$$W_i(v_i) - W_i(v_i') \leq \lambda_i(v_i)\big[v_i(f(v_i)) - v_i'(f(v_i))\big] \ \forall \ v_i, v_i' \in V_i. \tag{10}$$

The explicit construction of W_i involves construction of a weighted directed graph and finding shortest paths in such a graph - see Vohra (2011). From this, we can define $t_i : V_i \to \mathbb{R}$ as follows.

$$t_i(v_i) = \lambda_i(v_i)v_i(f(v_i)) - W_i(v_i) \ \forall \ v_i \in V_i.$$

PROPOSITION **2** *If λ_i makes f multiplier cycle monotone, then (λ_i, t_i) is an incentive compatible linear mechanism.*

Proof: Substituting in Inequality (10), we get for all $v_i, v_i' \in V_i$,

$$\lambda_i(v_i)v_i(f(v_i)) - t_i(v_i) - \lambda_i(v_i')v_i'(f(v_i')) + t_i(v_i') \leq \lambda_i(v_i)\big[v_i(f(v_i)) - v_i'(f(v_i))\big]$$

This gives us the desired incentive constraints: for all $v_i, v_i' \in V_i$

$$\lambda_i(v_i')v_i'(f(v_i')) - t_i(v_i') \geq \lambda_i(v_i)v_i'(f(v_i)) - t_i(v_i').$$

■

Remark. Consider a type space V_i and assume that there exists a type $\underline{v}_i \in V_i$ such that $\underline{v}_i(a) = 0$ for all $a \in A$. Further, assume that for every $v_i \in V_i$ and for every $a \in A$, we have $v_i(a) \geq 0$. In this type space, we can show that there is a linear mechanism that will be individually rational and the payments of agents will be non-negative. To see this, the W_i map constructed in the proof can be constructed such that $W_i(\underline{v}_i) = 0$ - this is easily done by translating any W_i map to a new map with $W_i(\underline{v}_i) = 0$. In that case, the net utility of agent i when his type is v_i is given by

$$\begin{aligned}
\lambda_i(v_i)v_i(f(v_i)) - t_i(v_i) &= W_i(v_i) \\
&\geq W_i(\underline{v}_i) - \lambda_i(\underline{v}_i)\big[\underline{v}_i(f(\underline{v}_i)) - v_i(f(\underline{v}_i))\big] \\
&= \lambda_i(\underline{v}_i)v_i(f(\underline{v}_i)) \\
&\geq 0.
\end{aligned}$$

Similarly, for any v_i,

$$\begin{aligned} W_i(v_i) &\leq W_i(\underline{v}_i) + \lambda_i(v_i)\big[v_i(f(v_i)) - \underline{v}_i(f(v_i))\big] \\ &= \lambda_i(v_i)v_i(f(v_i)). \end{aligned}$$

Hence, $t_i(v_i) \geq 0$. Finally, note that we can always scale (λ_i, t_i) such that λ_i lies between 0 and 1 while maintaining $t_i(\cdot) \geq 0$ and individual rationality. Hence, there are linear mechanisms in this type space where the payments of agents are non-negative and all agents are individually rational.

Proof of Theorem 2

We will show that if f is an AUM satisfying consistent tie-breaking, then it is acyclic. By Theorem 3, we will be done. Let P be the linear order on the set of alternatives that is used to consistently break ties in f. Further, let W be a monotone aggregate utility function such that $f(v) \in \operatorname{argmax}_{a \in A} W(a, v^a)$ for every $v \in V$.

Fix an agent $i \in N$ and type profile of other agents at v_{-i}. Consider a sequence of types $v_i^1 \succeq^f \ldots \succeq^f v_i^k$. Pick any $j \in \{1, \ldots, k-1\}$. Let $f(v_i^j, v_{-i}) = a_j$ and $f(v_i^{j+1}, v_{-i}) = a_{j+1}$. Since $v_i^j \succeq^f v_i^{j+1}$, we have $v_i^j(a_{j+1}) \geq v_i^{j+1}(a_{j+1})$. Denote the utility vector of any alternative c in type profile (v_i^j, v_{-i}) as $v^{j,c}$. By monotonicity of W, we get

$$W(a_{j+1}, v^{j,a_{j+1}}) \geq W(a_{j+1}, v^{(j+1),a_{j+1}}).$$

Since $f(v_i^j, v_{-i}) = a_j$, we have

$$W(a_j, v^{j,a_j}) \geq W(a_{j+1}, v^{j,a_{j+1}}).$$

Combining this with the previous inequality, we get

$$W(a_j, v^{j,a_j}) \geq W(a_{j+1}, v^{j,a_{j+1}}) \geq W(a_{j+1}, v^{(j+1),a_{j+1}}).$$

Using it over all $j \in \{1, \ldots, k-1\}$, we get that

$$\begin{aligned} W(a_1, v^{1,a_1}) &\geq W(a_2, v^{1,a_2}) \\ &\geq W(a_2, v^{2,a_2}) \\ &\geq \ldots \\ &\geq \ldots \\ &\geq W(a_k, v^{k-1,a_k}) \\ &\geq W(a_k, v^{k,a_k}) \end{aligned}$$

Since $f(v_i^k, v_{-i}) = a_k$, we know that $W(a_k, v^{k,a_k}) \geq W(a_1, v^{k,a_1})$. Hence, we get

$$W(a_1, v^{1,a_1}) \geq W(a_1, v^{k,a_1}). \tag{11}$$

Now, assume for contradiction that $v_i^k \succ^f v_i^1$. So, $v_i^k(a_1) > v_i^1(a_1)$. By monotonicity of W, we have $W(a_1, v^{k,a_1}) \geq W(a_1, v^{1,a_1})$. Using Inequality (11), we get

$$\begin{aligned} W(a_1, v^{1,a_1}) &= W(a_2, v^{1,a_2}) \\ &= W(a_2, v^{2,a_2}) \\ &= \ldots \\ &= \ldots \\ &= W(a_k, v^{k-1,a_k}) \\ &= W(a_k, v^{k,a_k}) \\ &= W(a_1, v^{k,a_1}) \\ &= W(a_1, v^{1,a_1}). \end{aligned}$$

Now, pick any $j \in \{1, \ldots, k-1\}$. Since $W(a_j, v^{j,a_j}) = W(a_{j+1}, v^{j,a_{j+1}})$, by consistent tie-breaking, it must be that either $a_j = a_{j+1}$ or $a_j P a_{j+1}$. Using it for all $j \in \{1, \ldots, k-1\}$, we see that either $a_1 = a_2 = \ldots = a_k$ or $a_1 P a_k$. But $W(a_k, v^{k,a_k}) = W(a_1, v^{k,a_1})$ implies that $a_1 P a_k$ is not possible. Hence, $a_1 = a_2 = \ldots = a_k = a$ for some $a \in A$. But this implies that $v_i^1(a) \geq v_i^2(a) \geq \ldots \geq v_i^k(a)$, and this contradicts that $v_i^k \succ^f v_i^1$.

Statement and Proof of Lemma 3

LEMMA **3** *Suppose the type space is countable. If an scf is acyclic, it can be implemented using a contingent contract.*

Proof: Again throughout the proof, we fix an agent i and the type profile of the other agents at v_{-i}. For notational simplicity, we will suppress the dependence on v_{-i}. Consider an scf f. We need to show that there exists a contingent contract s_i such that for every $v_i, v_i' \in V_i$, we have

$$s(v_i(f(v_i)), v_i) \geq s(v_i(f(v_i')), v_i'). \tag{12}$$

We will define an incomplete binary relation $\succ_s$, $\sim_s$ over tuples $\{v_i(f(v_i')), v_i'\}$ for all $v_i, v_i' \in V_i$. These tuples correspond to a type v_i making a report of v_i'. We first define the

relation $\succ_{s_0}$ and $\sim_s$ now.

$$\begin{array}{ll}
\{v_i(f(v_i')), v_i'\} \succ_{s_0} \{v_i'(f(v_i')), v_i'\} & \text{if } v_i(f(v_i')) > v_i'(f(v_i')) \\
\{v_i'(f(v_i')), v_i'\} \succ_{s_0} \{v_i(f(v_i')), v_i'\} & \text{if } v_i(f(v_i')) < v_i'(f(v_i')) \\
\{v_i(f(v_i')), v_i'\} \sim_s \{v_i'(f(v_i')), v_i'\} & \text{if } v_i(f(v_i')) = v_i'(f(v_i')) \\
\{v_i'(f(v_i')), v_i'\} \sim_s \{v_i(f(v_i')), v_i'\} & \text{if } v_i(f(v_i')) = v_i'(f(v_i')) \\
\{v_i(f(v_i)), v_i\} \succ_{s_0} \{v_i(f(v_i')), v_i'\} & \text{for all } v_i' \neq v_i
\end{array}$$

We define $\succ_s$ as the transitive closure of $\succ_{s_0}$. Formally, we say $\{v_i(f(v_i')), v_i'\} \succ_s \{\hat{v}_i(f(\hat{v}_i')), \hat{v}_i'\}$ if there exists a finite sequence $\{\{v_i^1(f(v_i'^1)), v_i'^1\}, \ldots, \{v_i^K(f(v_i'^K)), v_i'^K\}\}$ such that

$$\{v_i(f(v_i')), v_i'\} R_1 \{v_i^1(f(v_i'^1)), v_i'^1\} R_2 \cdots R_K \{v_i^K(f(v_i'^K)), v_i'^K\} R_{K+1} \{\hat{v}_i(f(\hat{v}_i')), \hat{v}_i'\}$$

where $R_k \in \{\succ_{s_0}, \sim_s\}$ and at least one $R_k \equiv \succ_{s_0}$. It is easy to argue that acyclicality of f implies that the relation $\succ_s$ is irreflexive.

Since $\succ_s$ is irreflexive and transitive and V_i is countable, we can then use a standard representation theorem (Fishburn, 1970) which guarantees the existence of a function s_i which respects $\succ_s$. ■

Proof of Theorem 5

$1 \Rightarrow 2$. Clearly, an AUM with consistent tie-breaking satisfies binary independence and it is implementable by Theorem 2.

$2 \Rightarrow 3$. This follows from Theorem 3.

$3 \Rightarrow 1$. We do this part of the proof in many steps. Let f be a 2-acyclic scf satisfying binary independence.

Step 1. We show that f satisfies the following positive association property. We say f satisfies **weak positive association (WPA)** if for every pair of type profiles v, v' with $f(v) = a$, $v_i'(a) \geq v_i(a)$ for all $i \in N$, $v_i'(x) = v_i(x)$ for all $x \neq a$, for all $i \in N$, we have $f(v') = a$.

To see this, consider two type profiles v and $(\bar{v}_i, v_{-i})$ with $f(v) = a$, $\bar{v}_i(a) > v_i(a)$, and $\bar{v}_i(x) = v_i(x)$ for all $x \neq a$. Assume for contradiction $f(\bar{v}_i, v_{-i}) = b \neq a$. So, we have $\bar{v}_i(a) > v_i(a)$ and $v_i(b) = \bar{v}_i(b)$, and this contradicts 2-acyclicity of f. By repeatedly applying this argument for all $i \in N$, we get that f satisfies WPA.

STEP 2. Let $\bar{A} := \{a \in A : \text{ there exists } v \in V \text{ such that } f(v) = a\}$, i.e., $\bar{A}$ is the range of f. Let $\bar{\mathcal{X}} := \{(a, x) \in \mathcal{X} : a \in \bar{A}\}$. Note that since V is finite, $\bar{A}$ (the range of f) is finite. As a result, $\bar{\mathcal{X}}$ is also finite. Now, we define a binary relation $\rhd^f$ on the elements of $\bar{\mathcal{X}}$. For any $(a, x), (b, y) \in \bar{\mathcal{X}}$ with $a \neq b$, we let

$$(a, x) \rhd^f (b, y) \text{ if there exists } v \in V \text{ such that } v^a = x, v^b = y, f(v) = a$$

and for any $(a, x), (a, x + \epsilon) \in \bar{\mathcal{X}}$ with $\epsilon \in \mathbb{R}^n_+$ and $x \neq (x + \epsilon)$, we let

$$(a, x + \epsilon) \rhd^f (a, x).$$

Note that the binary relation is only a partial order. Binary independence immediately implies that $\rhd^f$ is anti-symmetric. To see this, pick any $(a, x), (b, y) \in \bar{\mathcal{X}}$ with $a \neq b$. Let $(a, x) \rhd^f (b, y)$. This implies that there exists a type profile v with $v^a = x, v^b = y$, and $f(v) = a$. By binary independence, for any other v' with $v'^a = x, v'^b = y$, we have $f(v') \neq b$. Hence, $(b, y) \not\rhd^f (a, x)$.

STEP 3. We will say that the binary relation $\rhd^f$ satisfies the following monotonicity property. Pick distinct $a, b \in \bar{A}$ and $x \in \mathcal{U}^a, y \in \mathcal{U}^b$ such that $(a, x) \rhd^f (b, y)$. Then, there exists v such that $v^a = x, v^b = y$, and $f(v) = a$. Choose $\epsilon \in \mathbb{R}^n_+$ such that $(x + \epsilon) \in \mathcal{U}^a$. Since f satisfies WPA (Step 1), at profile v' with $v'^a = x + \epsilon$ and $v'^c = v^c$ for all $c \in A \setminus \{a\}$, we have $f(v') = a$ (note that such v' exists due to richness of type space). Hence, $(a, x + \epsilon) \rhd^f (b, y)$.

STEP 4. Finally, this implies that $\rhd^f$ is transitive. Suppose $a, b, c \in \bar{A}$ are three distinct alternatives and pick $(a, x), (b, y), (c, z) \in \bar{\mathcal{X}}$ such that $(a, x) \rhd^f (b, y) \rhd^f (c, z)$. Since $(a, x) \rhd^f (b, y)$, there exists a type profile v such that $v^a = x, v^b = y$, and $f(v) = a$. Note that this implies that $(a, x) \rhd^f (a', v^{a'})$ for all $a' \in \bar{A} \setminus \{a\}$. Consider a utility profile v', where $v'^c = z$ and $v'^{a'} = v^{a'}$ for all $a' \in A \setminus \{c\}$. Since $(a, x) \rhd^f (a', v'^{a'})$ for all $a' \in \bar{A} \setminus \{a, c\}$, $f(v') \in \{a, c\}$. If $f(v') = c$, then $(c, z) \rhd^f (b, y)$, which is a contradiction, since $\rhd^f$ is anti-symmetric (Step 2). Hence, $f(v') = a$, which implies that $(a, x) \rhd^f (c, z)$.

The other case is $(a, x + \epsilon) \rhd^f (a, x) \rhd^f (b, y)$ for some $\epsilon \in \mathbb{R}^n_+$ with $x \neq (x + \epsilon)$ and $x, (x + \epsilon) \in \mathcal{U}^a$. But by Step 3, $(a, x + \epsilon) \rhd^f (b, y)$.

Finally, the case $(b, y) \rhd^f (a, x+\epsilon) \rhd^f (a, x)$, where $\epsilon \in \mathbb{R}^n_+$ and $x \neq (x+\epsilon), x, (x+\epsilon) \in \mathcal{U}^a$. Since $(b, y) \rhd^f (a, x + \epsilon)$, there exists a profile v with $v^b = y, v^a = x + \epsilon$, and $f(v) = b$. This implies that $(b, y) \rhd^f (a', v^{a'})$ for all $a' \in \bar{A} \setminus \{b\}$. Now, consider the profile v' where $v'^a = x, v'^{a'} = v^{a'}$ for all $a' \neq a$ (by richness, such a type profile exists). By binary independence, $f(v') \in \{b, a\}$. If $f(v') = a$, then $(a, x) \rhd^f (b, y)$ and Step 3 implies that $(a, x + \epsilon) \rhd^f (b, y)$, which is a contradiction. Hence, $f(v') = b$, and this implies that

$(b, y) \rhd^f (a, x)$.

STEP 5. This shows that $\rhd^f$ is an irreflexive, anti-symmetric, transitive binary relation on $\bar{\mathcal{X}}$. By Szpilrajn's extension theorem, we can extend it to a complete, irreflexive, anti-symmetric, transitive binary relation on $\bar{\mathcal{X}}$. Since $\bar{\mathcal{X}}$ is finite, there is a utility representation $\bar{W} : \bar{\mathcal{X}} \to \mathbb{R}$ of this linear order. We can then extend this map to $W : \mathcal{X} \to \mathbb{R}$ as follows, for every $(a, x) \in \bar{\mathcal{X}}$, let $W(a, x) := \bar{W}(a, x)$. Then choose $\delta < \min_{(a,x)\in\bar{\mathcal{X}}} \bar{W}(a, x)$, and set $W(a, x) := \delta$ for every $(a, x) \notin \bar{\mathcal{X}}$.

Now, since $\rhd^f$ satisfies $(a, x + \epsilon) \rhd^f (a, x)$ for all $a \in \bar{A}$, for all $x, (x + \epsilon) \in \mathcal{U}^a$ with $\epsilon \in \mathbb{R}^n$ and $x \neq (x + \epsilon)$, W is monotone. Now, at every profile v, if $f(v) = a$, by definition, $(a, v^a) \rhd^f (b, v^b)$ for all $b \in \bar{A} \setminus \{a\}$, which implies that $W(a, v^a) > W(b, v^b)$ for all $b \neq a$. Hence, W is an AUM. Further, note that $\bar{W}$ is an injective map. Hence, no tie-breaking is necessary for W. So, vacuously, it is an AUM with consistent tie-breaking.

Appendix 2: Extensions

Throughout this appendix, we conduct the analysis for an arbitrary agent i, fix $v_{-i} \in V_{-i}$ and for notational convenience, we suppress the dependence on v_{-i}. Recall that we can do so because the incentive compatibility requirement is for each agent i and all possible reports v_{-i} of the other agents.

Partially Contractible Utilities

We note that in many occasions the entire utility may not be contractible. However, our results will continue to hold in some such situations. Suppose the utility of an agent has two components - (1) a revenue component, which is contractible and (2) a *happiness* component, which is not contractible. We assume that the happiness is a monotone function of the revenue and the alternative chosen. Formally, type v_i now reflects the *revenue* of agent i over various alternatives and this is contractible.

There is a map

$$g_i : \mathbb{R} \times A \to \mathbb{R}$$

that gives the **non-contractible utility** of agent i. We assume that g_i is **non-decreasing** in the first argument.

Consider an scf f. Given a contingent contract s_i, the net utility of agent i by reporting v_i' with true type v_i is given by

$$s_i(v_i(f(v_i')), v_i') + g_i(v_i(f(v_i')), f(v_i')).$$

Similarly, given a linear contract (r_i, t_i), the net utility of agent i by reporting v_i' with true type v_i is given by

$$r_i(v_i')v_i(f(v_i')) + g_i(v_i(f(v_i')), f(v_i')) - t_i(v_i').$$

We will show that Theorem 3 continues to hold even under this setting. Of course, Theorem 1 does not hold any longer since we there are components of utility that are not contractible. Since Theorem 3 continues to hold, with an appropriate redefinition of aggregate utility maximizers, we can also show that Theorem 2 holds.

As before, for any scf f, we define the binary relation $\succ^f$ as follows. For any $v_i, v_i' \in V_i$, we say $v_i \succ^f v_i'$ if $v_i(f(v_i')) > v_i'(f(v_i'))$. We also define the binary relation $\succeq^f$ as follows. For any $v_i, v_i' \in V_i$, we say $v_i \succeq^f v_i'$ if $v_i(f(v_i')) \geq v_i'(f(v_i'))$.

Definition **13** *An scf f is acyclic if for any sequence of types $v_i^1, \ldots, v_i^k$ with $v_i^1 \succeq^f v_i^2 \succeq^f \ldots \succeq^f v_i^k$, we have $v_i^k \not\succ^f v_i^1$.*

As before, we can show the necessity of acyclicity.

LEMMA **4** *If an scf is implementable by a contingent contract, then it is acyclic.*

Proof: Suppose scf f is implementable by a contingent contract s_i. Consider any sequence of types $v_i^1, \ldots, v_i^k$ with $v_i^1 \succeq^f v_i^2 \succeq^f \ldots \succeq^f v_i^k$. Choose $j \in \{1, \ldots, k-1\}$. Since f is implementable by s_i, we get that

$$\begin{aligned} s_i(v_i^j(f(v_i^j)), v_i^j) + g_i(v_i^j(f(v_i^j)), f(v_i^j)) &\geq s_i(v_i^j(f(v_i^{j+1})), v_i^{j+1}) + g_i(v_i^j(f(v_i^{j+1})), f(v_i^{j+1})) \\ &\geq s_i(v_i^{j+1}(f(v_i^{j+1})), v_i^{j+1}) + g_i(v_i^{j+1}(f(v_i^{j+1})), f(v_i^{j+1})) \end{aligned}$$

where the second inequality used the fact that $v_i^j \succeq^f v^{j+1}$, s_i is increasing in the first argument, and g_i is non-decreasing in the first argument. Hence, we get that for any $j \in \{1, \ldots, k-1\}$, we have

$$s_i(v_i^j(f(v_i^j)), v_i^j) + g_i(v_i^j(f(v_i^j)), f(v_i^j)) \geq s_i(v_i^{j+1}(f(v_i^{j+1})), v_i^{j+1}) + g_i(v_i^{j+1}(f(v_i^{j+1})), f(v_i^{j+1})). \tag{13}$$

Adding Inequality (13) for all $j \in \{1, \ldots, k-1\}$ and telescoping, we get

$$s_i(v_i^1(f(v_i^1)), v_i^1) + g_i(v_i^1(f(v_i^1)), f(v_i^1)) \geq s_i(v_i^k(f(v_i^k)), v_i^k) + g_i(v_i^k(f(v_i^k)), f(v_i^k)). \tag{14}$$

Since f is implementable, we have $s_i(v_i^k(f(v_i^k)), v_i^k) + g_i(v_i^k(f(v_i^k)), f(v_i^k)) \geq s_i(v_i^k(f(v_i^1)), v_i^1) + g_i(v_i^k(f(v_i^1)), f(v_i^1))$. This along with Inequality (14) gives us

$$s_i(v_i^1(f(v_i^1)), v_i^1) + g_i(v_i^1(f(v_i^1)), f(v_i^1)) \geq s_i(v_i^k(f(v_i^1)), v_i^1) + g_i(v_i^k(f(v_i^1)), f(v_i^1)). \tag{15}$$

Now, assume for contradiction, $v_i^k \succ^f v_i^1$. Then, $v_i^k(f(v_i^1)) > v_i^1(f(v_i^1))$. Since s_i is strictly increasing in the first argument and g_i is non-decreasing in the first argument, we get that

$$s_i(v_i^k(f(v_i^1)), v_i^1) + g_i(v_i^k(f(v_i^1)), f(v_i^1)) > s_i(v_i^1(f(v_i^1)), v_i^1) + g_i(v_i^1(f(v_i^1)), f(v_i^1)). \tag{16}$$

This is a contradiction to Inequality (15). ■

We now proceed to show that the remainder of the proof of Theorem 3 can be adapted straightforwardly. First, we define some terminology. For any $v_i, v_i' \in V_i$, let

$$d(v_i, v_i') := v_i(f(v_i)) - v_i'(f(v_i))$$

and

$$d'(v_i, v_i') := g_i(v_i(f(v_i)), f(v_i)) - g_i(v_i'(f(v_i)), f(v_i)).$$

DEFINITION **14** *An scf f is* **generalized multiplier cycle monotone** *if there exists $\lambda_i : V_i \to (0,\infty)$ such that for every sequence of types $(v_i^1, \ldots, v_i^k, v_i^{k+1} \equiv v_i^1)$ we have*

$$\sum_{j=1}^{k} \left[\lambda_i(v_i^j) d(v_i^j, v_i^{j+1}) + d'(v_i^j, v_i^{j+1})\right] \geq 0.$$

PROPOSITION **3** *An scf f is implementable by a linear contract if and only if it is generalized multiplier cycle monotone.*

Proof: The necessity of generalized multiplier cycle monotonicity follows by adding any cycle of incentive constraints. For sufficiency, suppose f satisfies generalized multiplier cycle monotonicity. Let $\lambda_i : V_i \to (0,\infty)$ be the corresponding multiplier. Then, by the Rochet-Rockefellar theorem, there exists a map $W : V_i \to \mathbb{R}$ such that for every $v_i, v_i' \in V_i$, we have

$$W(v_i) - W(v_i') \leq \left[\lambda_i(v_i) d(v_i, v_i') + d'(v_i, v_i')\right]. \tag{17}$$

Now, for any $v_i \in V_i$, let

$$t_i(v_i) := \lambda_i(v_i) v_i(f(v_i)) + g_i(v_i(f(v_i)), f(v_i)) - W(v_i).$$

Now, substituting in Inequality (17), we get for every $v_i, v_i' \in V_i$,

$$\begin{aligned} W(v_i) - W(v_i') &= \lambda_i(v_i) v_i(f(v_i)) + g_i(v_i(f(v_i)), f(v_i)) - t_i(v_i) \\ &\quad - \lambda_i(v_i') v_i'(f(v_i')) - g_i(v_i'(f(v_i')), f(v_i')) + t_i(v_i') \\ &\leq \lambda_i(v_i) v_i(f(v_i)) - \lambda_i(v_i) v_i'(f(v_i)) + g_i(v_i(f(v_i)), f(v_i)) - g_i(v_i'(f(v_i)), f(v_i)). \end{aligned}$$

Canceling terms, we get

$$\lambda_i(v_i') v_i'(f(v_i')) + g_i(v_i'(f(v_i')), f(v_i')) - t_i(v_i') \geq \lambda_i(v_i) v_i'(f(v_i)) + g_i(v_i'(f(v_i)), f(v_i)) - t_i(v_i).$$

This gives us the desired incentive constraints. ■

We will now show that if f is acyclic, then it is generalized multiplier cycle monotone. To do so, we first observe that if f is acyclic, then we can apply Lemma 2 to claim that the type space can be f-ordered-partitioned. Now, we can use this to construct a $\lambda_i : V_i \to (0,\infty)$ map recursively. Let $(V_i^1, \ldots, V_i^K)$ be an f-ordered-partition of V_i. First, we set for all $v_i \in V_i^K$,

$$\lambda_i(v_i) := 1.$$

Having defined $\lambda_i(v_i)$ for all $v_i \in V_i^{k+1} \cup \ldots \cup V_i^K$, we define $\lambda_i(v_i)$ for all $v_i \in V_i^k$. Let C be any cycle of types $(v_i^1, \ldots, v_i^q, v_i^1)$ involving types in $V_i^k \cup \ldots \cup V_i^K$ with at least one type in

V_i^k and at least one type in $V_i^{k+1} \cup \ldots \cup V_i^K$. Let $\mathcal{C}$ be the set of all such cycles. Now, define for each cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1) \in \mathcal{C}$,

$$L(C) := \sum_{v_i^j \in C \cap (V_i^{k+1} \cup \ldots \cup V_i^K)} \lambda_i(v_i^j) d(v_i^j, v_i^{j+1}) + \sum_{j=1}^{q} d'(v_i^j, v_i^{j+1}).$$

and

$$\ell(C) := \sum_{v_i^j \in V_i^k \cap C} d(v_i^j, v_i^{j+1}).$$

Now, consider two possible cases.

- If $L(C) \geq 0$ for all $C \in \mathcal{C}$, then set $\lambda_i(v_i) = 1$ for all $v_i \in V_i^k$.
- If $L(C) < 0$ for some $C \in \mathcal{C}$, we proceed as follows. Since V_i is f-ordered-partitioned, for every $v_i \in V_i^k$ and $v_i' \in (V_i^{k+1} \cup \ldots \cup V_i^K)$, we have $d(v_i, v_i') > 0$ (Property P2). Similarly, for every $v_i, v_i' \in V_i^k$, we have $d(v_i, v_i') \geq 0$ (Property P1). Then, for every $C \in \mathcal{C}$, we must have $\ell(C) > 0$ since it involves at least one type from V_i^k and at least one type from $(V_i^{k+1} \cup \ldots \cup V_i^K)$. Now, for every $v_i \in V_i^k$, define

$$\lambda_i(v_i) := \max_{C \in \mathcal{C}} \frac{-L(C)}{\ell(C)}.$$

We thus recursively define the λ_i map.

PROPOSITION 4 *If f is acyclic, then λ_i makes f generalized multiplier cycle monotone.*

Proof: Consider any cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1)$. We will show that

$$\sum_{j=1}^{q} \lambda_i(v_i^j) d(v_i^j, v_i^{j+1}) + d'(v_i^j, v_i^{j+1}) \geq 0. \tag{18}$$

If $C \subseteq V_i^K$, then $d(v_i^j, v_i^{j+1}) \geq 0$, $d'(v_i^j, v_i^{j+1}) \geq 0$, and $\lambda_i(v_i^j) = \lambda_i(v_i^{j+1})$ for all $v_i^j, v_i^{j+1} \in C$. Hence, Inequality (18) holds. Now, suppose Inequality (18) is true for all cycles $C \subseteq (V_i^{k+1} \cup \ldots \cup V_i^K)$. Consider a cycle $C \equiv (v_i^1, \ldots, v_i^q, v_i^{q+1} \equiv v_i^1)$ involving types in $(V_i^k \cup \ldots \cup V_i^K)$. If each type in C is in V_i^k, then again $d(v_i^j, v_i^{j+1}) \geq 0$, $d'(v_i^j, v_i^{j+1}) \geq 0$, and $\lambda_i(v_i^j) = \lambda_i(v_i^{j+1})$ for all $v_i^j, v_i^{j+1} \in C$. Hence, Inequality (18) holds. By our hypothesis, if all types in C belong to $(V_i^{k+1} \cup \ldots V_i^K)$, then again Inequality (18) holds. So, assume that C is a cycle that involves at least one type from V_i^k and at least one type from $(V_i^{k+1} \cup \ldots \cup V_i^K)$. Let $\lambda_i(v_i) = \mu$ for all $v_i \in V_i^k$. By definition,

$$\sum_{v_i^j \in C} \left[\lambda_i(v_i^j) d(v_i^j, v_i^{j+1}) + d'(v_i^j, v_i^{j+1})\right] = L(C) + \mu \ell(C) \geq 0,$$

where the last inequality followed from the definition of μ. Hence, Inequality (18) again holds. Proceeding like this inductively, we complete the proof. ■

To summarize, we have shown the following result.

THEOREM **6** *Consider the partially contractible environment and suppose the type space is finite. Then, for any scf f, the following are equivalent.*

1. *f is implementable by a contingent contract.*
2. *f is acyclic.*
3. *f is generalized multiplier cycle monotone.*
4. *f is implementable by a linear contract.*

An Infinite Type Space where the Equivalence Holds

We show that Theorem 3 extends to a model with infinite set of types under additional conditions. We make the following assumptions.

A1 The set of alternatives A is a metric space.

A2 The set of types V_i is a compact metric space and each $v_i \in V_i$ is continuous in a.

A3 The scf $f(\cdot)$ is continuous in v_i.

THEOREM **7** *Let f be an scf and assumptions A1-A3 hold. If scf f can be implemented by a contingent contract s_i which is twice continuously (partially) differentiable in the first argument, then f can also be implemented by a linear contract.*

Proof: We are given that f can be implemented by a contingent contract $s_i : \mathbb{R} \times V_i \to \mathbb{R}$ which is twice continuously differentiable in the first argument. We first show that this implies that f can also be implemented by a contingent contract $\tilde{s}_i$ which is convex in the first argument. Consider the following transformation of s:

$$\tilde{s}_i = e^{\gamma s_i} \text{ where } \gamma > 0.$$

Clearly, since $\tilde{s}_i$ is a monotone transformation of s_i, it is both strictly increasing in the first argument and incentive compatible. Therefore, it also implements f. We denote partial derivatives of with respect to the first argument by $\frac{\partial}{\partial u_i}$.

Since s_i is twice differentiable in the first argument, so is $\tilde{s}_i$ and its second partial derivative is given by

$$\frac{\partial^2 \tilde{s}_i}{\partial u_i^2} = \gamma e^{\gamma s_i} \left(\frac{\partial \tilde{s}_i}{\partial u_i} \right)^2 \left(\frac{\partial^2 \tilde{s}_i / \partial u_i^2}{(\partial \tilde{s}_i / \partial u_i)^2} + \gamma \right).$$

Now, since V_i is compact, f is continuous and s_i is twice continuously differentiable in the first argument, this implies that

$$\frac{\partial^2 \tilde{s}_i(v_i(f(v_i')), v_i')/\partial u_i^2}{(\partial \tilde{s}(v_i(f(v_i')), v_i')/\partial u_i)^2} \text{ is bounded from below for all } v_i, v_i' \in V_i.$$

This is because the above function is continuous on the compact set $V_i \times V_i$ and hence must attain a minimum.

This in turn implies that there exists a large and finite $\gamma > 0$ such that $\tilde{s}_i$ is convex in the first argument. Incentive compatibility and convexity of $\tilde{s}_i$ applied in turn then yield the following inequality for all $v_i, v_i' \in V_i$

$$\begin{aligned} \tilde{s}_i(v_i(f(v_i)), v_i) &\geq \tilde{s}_i(v_i(f(v_i')), v_i') \\ &\geq \tilde{s}_i(v_i'(f(v_i')), v_i') + \frac{\partial \tilde{s}(v_i'(f(v_i')), v_i')}{\partial u_i} [v_i(f(v_i')) - v_i'(f(v_i'))]. \end{aligned}$$

Now set multipliers $\lambda_i(v_i') = \frac{\partial \tilde{s}_i(v_i'(f(v_i')), v_i')}{\partial u_i} > 0$ for all $v_i' \in V_i$ and notice that f will satisfy multiplier cycle monotonicity with these multipliers. Of course, this implies that f can be implemented by a linear contract (Proposition 2) which completes the proof. ■

Sufficiency of 2-acyclicity in a Linear One Dimensional Model

In this section, we describe a simple model of a one dimensional type space with uncountable types, where 2-acyclicity is sufficient .

We assume that the set of alternatives A is finite. Additionally, we assume that types are one dimensional and linear. Formally, for every alternative $a \in A$, there exists $\kappa_a \geq 0$ and γ_a such that for all i

$$v_i(a) = \kappa_a v_i + \gamma_a \qquad \text{where } v_i \in V_i \subseteq \mathbb{R}.$$

The following is the characterization result.

THEOREM **8** *Suppose A is finite and the types are one dimensional and linear. Then, the following conditions on an scf f are equivalent.*

1. *f satisfies 2-acyclicity.*

2. *f is multiplier 2-cycle monotone.*

3. *f is implementable by a linear contract.*

4. *f is implementable by a contingent contract.*

Proof: $1 \Rightarrow 2$. Define the map $\nu : A \to \mathbb{R}_+$ as follows. For every $a \in A$,

$$\nu(a) = \begin{cases} \frac{1}{\kappa_a} & \text{if } \kappa_a \neq 0 \\ 0 & \text{if } \kappa_a = 0 \end{cases}$$

Further, define $\nu^* := \max_{a \in A} \nu(a)$ and $V_i^0 := \{v_i \in V_i : \kappa_{f(v_i)} = 0\}$. Now, define $r_i : V_i \to (0, 1]$ as follows. Fix an $\epsilon \in (0, 1]$. For every $v_1 \in V_i$,

$$r_i(v_i) = \begin{cases} \epsilon & \forall\ v_i \in V_i^0 \\ \frac{\nu(f(v_i))}{\nu^*} & \forall\ v_i \in V \setminus V_i^0. \end{cases}$$

Now, note that if $v_i \in V_i^0$, then $r_i(v_i)\kappa_{f(v_i)} = 0$ and if $v_i \in V \setminus V_i^0$, then $r_i(v_i)\kappa_{f(v_i)} = \frac{1}{\nu^*}$. Hence, for every $v \in V_i^0$ and $v' \in V \setminus V_i^0$, we have

$$r_i(v_i')\kappa_{f(v_i')} > r_i(v)\kappa_{f(v_i)}. \tag{19}$$

Now, consider any $v_i, v_i' \in V$. Since f is 2-acyclic, it means $v_i' \succeq^f v_i$ implies $v_i \not\succ^f v_i'$. Equivalently, $(v_i' - v_i)\kappa_{f(v_i)} \geq 0$ implies $(v_i' - v_i)\kappa_{f(v_i')} \geq 0$. Equivalently, if $v_i > v_i'$ and $\kappa_{f(v_i)} = 0$ then $\kappa_{f(v_i')} = 0$. This further means, if $v_i \in V_i^0$ and $v_i' < v_i$, then $v_i' \in V_i^0$. Hence, using Inequality (19), we get that if $v_i' > v_i$, then

$$r_i(v_i')\kappa_{f(v_i')} \geq r_i(v_i)\kappa_{f(v_i)}. \tag{20}$$

Now, for any $v_i, v_i' \in V$ with $v_i' > v_i$, multiplier 2-cycle monotonicity requires that

$$(v_i' - v_i)\big(r_i(v_i')\kappa_{f(v_i')} - r_i(v_i)\kappa_{f(v_i)}\big) \geq 0. \tag{21}$$

This is true because of Inequality (20).

$2 \Rightarrow 3$. Using Proposition 2, it is enough to show that if f is multiplier 2-cycle monotone, then it is multiplier cycle monotone. Because f satisfies multiplier 2-cycle monotonicity, for any $v_i' > v_i$, Inequality (21) is satisfied. But, this implies that Inequality (20) is satisfied.

Assume for contradiction that f fails multiplier cycle monotonicity. Let k be the smallest integer such that f fails multiplier k-cycle monotonicity. Since f satisfies multiplier 2-cycle

monotonicity, $k \geq 3$. This means for every $r_i : V_i \to (0,1]$ and for some finite sequence of types $(v_i^1, \ldots, v_i^k)$, we have

$$\sum_{j=1}^{k} \ell^{f,r_i}(v_i^j, v_i^{j+1}) < 0,$$

where $v_i^{k+1} \equiv v_i^1$ and $\ell^{f,r_i}(v_i^j, v_i^{j+1}) := r_i(v_i^j)\big[v_i^j(f(v_i^j)) - v_i^{j+1}(f(v_i^j))\big]$. Consider a $r_i : V_i \to (0,1]$. Let $v_i^j > v_i^p$ for all $p \in \{1, \ldots, k\}\backslash\{j\}$. We will show that $\ell^{f,r_i}(v_i^{j-1}, v_i^j) + \ell^{f,r_i}(v_i^j, v_i^{j+1}) - \ell^{f,r_i}(v_i^{j-1}, v_i^{j+1}) \geq 0$. To see this,

$$\begin{aligned}
\ell^{f,r_i}(v_i^{j-1}, v_i^j) + \ell^{f,r_i}(v_i^j, v_i^{j+1}) - \ell^{f,r_i}(v_i^{j-1}, v_i^{j+1}) &= v_i^j[r_i(v_i^j)\kappa_{f(v_i^j)} - r_i(v_i^{j-1})\kappa_{f(v_i^{j-1})}] \\
&\quad + v_i^{j+1}[r_i(v_i^{j+1})\kappa_{f(v_i^{j+1})} - r_i(v_i^j)\kappa_{f(v_i^j)}] \\
&\quad - v_i^{j+1}[r_i(v_i^{j+1})\kappa_{f(v_i^{j+1})} - r_i(v_i^{j-1})\kappa_{f(v_i^{j-1})}] \\
&= (v_i^j - v_i^{j+1})[r_i(v_i^j)\kappa_{f(v_i^j)} - r_i(v_i^{j-1})\kappa_{f(v_i^{j-1})}] \\
&\geq 0,
\end{aligned}$$

where the last inequality follows from the fact that $v_i^j > v_i^{j+1}$ and applying Inequality (20). Since f satisfies multiplier $(k-1)$-cycle monotonicity, we know that $\ell^{f,r_i}(v_i^1, v_i^2) + \ldots + \ell^{f,r_i}(v_i^{j-2}, v_i^{j-1}) + \ell^{f,r_i}(v_i^{j-1}, v_i^{j+1}) + \ell^{f,r_i}(v_i^{j+1}, v_i^{j+2}) + \ldots + \ell^{f,r_i}(v_i^k, v_i^1) \geq 0$. But, because of the last inequality, we must have

$$\sum_{j=1}^{k} \ell^{f,r_i}(v_i^j, v_i^{j+1}) \geq 0,$$

which gives us a contradiction.

Of course, $3 \Rightarrow 4$ and Lemma 1 establishes that $4 \Rightarrow 1$. This concludes the proof. ■

Remark. A closer look at the proof of Theorem 8 reveals that if $\kappa_a > 0$ for all $a \in A$, then for every scf f, $V_i^0 = \emptyset$, and hence, every scf f satisfies 2-acyclicity vacuously. Thus, *every* scf can be implemented using a linear contract.

Sufficiency of 2-acyclicity in a Linear Two Dimensional Model

In this section, we consider a linear two dimensional generalization of the model in the previous section. Formally, for every alternative $a \in A$, there exists $\kappa_{a_1} \geq 0$, $\kappa_{a_2} > 0$ and γ_a such that for all i

$$v_i(a) = v_{i_1}\kappa_{a_1} + v_{i_2}\kappa_{a_2} + \gamma_a \qquad \text{where } v_i \in V_i \subseteq \mathbb{R}.$$

The proof will use the following normalized vector for an alternative $a \in A$, which we denote by

$$\kappa_a = \left(\frac{\kappa_{a_1}}{\kappa_{a_2}}, 1 \right).$$

Note that since we have restricted $\kappa_{a_2} > 0$,[13] the above normalized vector is well defined.

The next result shows that 2-acyclicity implies acyclicity in the linear two dimensional environment. Thus, from Lemma 3, we can conclude that in countable type spaces, 2-acyclicity is sufficient for implementability.

THEOREM **9** *In the linear two dimensional environment, 2-acyclicity implies acyclicity.*

Proof: We need to show that 2-acyclicity implies k-acyclicity for all k. We will proceed by induction on k. The base case of $k = 2$ is trivially true. As the induction hypothesis, we assume the implication holds for some $k > 2$. We will now show the induction step that 2-acyclicity implies $k + 1$-acyclicity.

Suppose f is 2-acyclic. Consider a sequence $v_i^1, \ldots, v_i^{k+1}$ with the following properties. For all $j \in \{1, \ldots, k-1\}$, each element is weakly greater than the succeeding and no element is strictly greater than any previous element in the sequence. Formally,

$$v_i^j \succeq v_i^{j+1} \text{ and } v_i^{j+1} \not\succ v_i^{j'} \text{ for all } j' \in \{1, \ldots, j\}$$

which is equivalent to

$$\kappa_{f(v_i^{j+1})}(v_i^j - v_i^{j+1}) \geq 0 \text{ and } \kappa_{f(v_i^{j'})}(v_i^{j+1} - v_i^{j'}) \leq 0 \text{ for all } j' \in \{1, \ldots, j\}.$$

Additionally, without loss of generality, we can take the inequality to be strict for v_i^1:

$$v_i^1 \succ v_i^2 \text{ or that } \kappa_{f(v_i^2)}(v_i^1 - v_i^2) > 0.$$

The induction hypothesis (k-acyclicity) then implies that

$$v_i^{j'} \not\succeq v_i^1 \text{ or that } \kappa_{f(v_i^1)}(v_i^{j'} - v_i^1) < 0 \text{ for all } j' \in \{2, \ldots, k\}.$$

Finally, v_i^{k+1} is such that

$$v_i^k \succeq v_i^{k+1} \text{ or } \kappa_{f(v_i^{k+1})}(v_i^k - v_i^{k+1}) \geq 0.$$

The induction hypothesis implies that

$$v_i^{k+1} \not\succ v_i^{j'} \text{ for all } j' \in \{2, \ldots, k\} \text{ or } \kappa_{f(v_i^{j'})}(v_i^{k+1} - v_i^{j'}) \leq 0 \text{ for all } j \in \{2, \ldots, k\}.$$

[13] This assumption is required for the result. It is possible to construct a simple counter example if we allow $\kappa_{a_2} = 0$.

It is sufficient to show that for such sequences it must be that

$$v_i^{k+1} \not\succeq v_i^1 \text{ or } \kappa_{f(v_i^1)}(v_i^{k+1} - v_i^1) < 0.$$

We consider two cases depending on how the first component of the normalized vector $\kappa_{f(v_i^1)}$ compares to the first components of the vectors $\kappa_{f(v_i^j)}$ for $j \in \{2, \ldots, k+1\}$.
Case I: The first component of $\kappa_{f(v_i^1)}$ is the largest or smallest in the sequence.

This implies that either (i) the first component of $\kappa_{f(v_i^{k+1})}$ lies between the first components of $\kappa_{f(v_i^1)}$ and $\kappa_{f(v_i^2)}$ or that (ii) the first component $\kappa_{f(v_i^2)}$ lies between the first component of $\kappa_{f(v_i^1)}$ and $\kappa_{f(v_i^{k+1})}$.

Consider subcase (i) first. Here there must be an $\alpha \in [0,1]$ such that $\kappa_{f(v_i^{k+1})} = \alpha\kappa_{f(v_i^1)} + (1-\alpha)\kappa_{f(v_i^2)}$. Then, it must be that $v_i^1 \succ v_i^{k+1}$ which can be seen from the following series of inequalities

$$\begin{aligned}
\kappa_{f(v_i^{k+1})}(v_i^1 - v_i^{k+1}) &= \kappa_{f(v_i^{k+1})}(v_i^1 - v_i^k) + \kappa_{f(v_i^{k+1})}(v_i^k - v_i^{k+1}) \\
&\geq (\alpha\kappa_{f(v_i^1)} + (1-\alpha)\kappa_{f(v_i^2)})(v_i^1 - v_i^k) \\
&= \alpha\kappa_{f(v_i^1)}(v_i^1 - v_i^k) + (1-\alpha)\kappa_{f(v_i^2)}(v_i^1 - v_i^2) + (1-\alpha)\kappa_{f(v_i^2)}(v_i^2 - v_i^k) \\
&> 0.
\end{aligned}$$

Note that the strictness follows from the fact that either or both of the first two terms in the above must be strictly positive depending on the value of α. But then applying 2-acyclicity to the sequence $\{v_i^1, v_i^{k+1}\}$ implies that $v_i^{k+1} \not\succeq v_i^1$.

Now consider subcase (ii). Here there must be an $\alpha \in [0,1]$ such that $\kappa_{f(v_i^2)} = \alpha\kappa_{f(v_i^1)} + (1-\alpha)\kappa_{f(v_i^{k+1})}$. Observe that

$$\kappa_{f(v_i^2)}(v_i^1 - v_i^{k+1}) = \kappa_{f(v_i^2)}(v_i^1 - v_i^2) + \kappa_{f(v_i^2)}(v_i^2 - v_i^{k+1}) > 0,$$

which in turn implies that

$$\alpha\kappa_{f(v_i^1)}(v_i^1 - v_i^{k+1}) + (1-\alpha)\kappa_{f(v_i^{k+1})}(v_i^1 - v_i^{k+1}) > 0.$$

Hence, it must be that either $\kappa_{f(v_i^{k+1})}(v_i^1 - v_i^{k+1}) \leq 0$ and $\kappa_{f(v_i^1)}(v_i^1 - v_i^{k+1}) > 0$ in which case this subcase is completed or that $\kappa_{f(v_i^{k+1})}(v_i^1 - v_i^{k+1}) > 0$. In the latter case, we can once again apply 2-acyclicity to the sequence $\{v_i^1, v_i^{k+1}\}$ and get the desired relation $v_i^{k+1} \not\succeq v_i^1$.
Case II: The ratio of the components in $\kappa_{f(v_i^1)}$ lies between some $\kappa_{f(v_i^j)}$ and $\kappa_{f(v_i^{j+1})}$ where $j \in \{2, \ldots, k\}$. Then, there must be an $\alpha \in [0,1]$ such that $\kappa_{f(v_i^1)} = \alpha\kappa_{f(v_i^j)} + (1-\alpha)\kappa_{f(v_i^{j+1})}$.

Then

$$\begin{aligned}
\kappa_{f(v_i^1)}(v_i^1 - v_i^{k+1}) &= \kappa_{f(v_i^1)}(v_i^1 - v_i^j) + \kappa_{f(v_i^1)}(v_i^j - v_i^{k+1}) \\
&> (\alpha\kappa_{f(v_i^j)} + (1-\alpha)\kappa_{f(v_i^{j+1})})(v_i^j - v_i^{k+1}) \\
&\geq (1-\alpha)\kappa_{f(v_i^{j+1})}(v_i^j - v_i^{k+1}) \\
&= (1-\alpha)\kappa_{f(v_i^{j+1})}(v_i^j - v_i^{j+1}) + (1-\alpha)\kappa_{f(v_i^{j+1})}(v_i^{j+1} - v_i^{k+1}) \\
&\geq 0
\end{aligned}$$

which completes the proof. ■

References

Abhishek, V., B. Hajek, and S. R. Williams (2012): "On Bidding with Securities: Risk aversion and Positive Dependence," *arXiv preprint arXiv:1111.1453*.

Afriat, S. N. (1967): "The Construction of Utility Functions from Expenditure Data," *International Economic Review*, 8, 67–77.

Arrow, K. (1979): "?The Property Rights Doctrine and Demand Revelation under Incomplete Information," in *Economics and human welfare*, New York Academic Press.

Ashlagi, I., M. Braverman, A. Hassidim, and D. Monderer (2010): "Monotonicity and Implementability," *Econometrica*, 78, 1749–1772.

Bikhchandani, S., S. Chatterji, R. Lavi, A. Mualem, N. Nisan, and A. Sen (2006): "Weak Monotonicity Characterizes Deterministic Dominant Strategy Implementation," *Econometrica*, 74, 1109–1132.

Brown, D. J. and C. Calsamiglia (2007): "The Nonparametric Approach to Applied Welfare Analysis," *Economic Theory*, 31, 183–188.

Carbajal, J. C., A. McLennan, and R. Tourky (2013): "Truthful Implementation and Preference Aggregation in Restricted Domains," *Forthcoming, Journal of Economic Theory*.

Che, Y.-K. and J. Kim (2010): "Bidding with Securities: Comment," *American Economic Review*, 100, 1929–1935.

Cramton, P., R. Gibbons, and P. Klemperer (1987): "Dissolving a Partnership Efficiently," *Econometrica*, 55, 615–632.

Crémer, J. (1987): "Auctions with contingent payments: Comment," *American Economic Review*, 77, 746.

d'Aspremont, C. and L.-A. Gérard-Varet (1979): "Incentives and Incomplete Information," *Journal of Public economics*, 11, 25–45.

d'Aspremont, C. and L. Gevers (2002): *Handbook of Social Choice and Welfare - Volume 2*, Elsevier, Amsterdam, chap. Social Welfare Functionals and Interpersonal Comparability (Chapter 10), 459–541, ediors: Kenneth J. Arrow, Amartya K. Sen, and K. Suzumura.

DeMarzo, P. M., I. Kremer, and A. Skrzypacz (2005): "Bidding with Securities: Auctions and Security Design," *American Economic Review*, 95, 936–959.

Fishburn, P. C. (1970): *Utility Theory for Decision Making*, John Wiley and Sons, New York.

Gorbenko, A. and A. Malenko (2010): "Competition Among Sellers in Securities Auctions," *American Economic Review*, 101, 1806–1841.

Green, J. R. and J.-J. Laffont (1979): *Incentives in Public Decision Making*, North-Holland.

Groves, T. (1973): "Incentives in Teams," *Econometrica*, 41, 617–631.

Guo, M. and V. Conitzer (2009): "Worst-case Optimal Redistribution of VCG Payments in Multi-unit Auctions," *Games and Economic Behavior*, 67, 69–98.

Hansen, R. G. (1985): "Auctions with Contingent Payments," *American Economic Review*, 75, 862–865.

Kos, N. and M. Messner (2013): "Extremal Incentive Compatible Transfers," *Journal of Economic Theory*, 148, 134–164.

Krishna, V. (2009): *Auction Theory*, Academic Press.

Laffont, J.-J. and J. Tirole (1986): "Using Cost Observation to Regulate Firms," *Journal of Political Economy*, 614–641.

Maskin, E. (1992): *Privatization*, Mohr, Tubingen, chap. Auctions and Privatization, 115–136, editor: Horst Siebert.

Mezzetti, C. (2004): "Mechanism Design with Interdependent Valuations: Efficiency," *Econometrica*, 72, 1617–1626.

Milgrom, P. R. and R. J. Weber (1982): "A Theory of Auctions and Competitive Bidding," *Econometrica*, 50, 1089–1122.

Mishra, D. and S. Roy (2013): "Implementation in Multidimensional Dichotomous Domains," *Theoretical Economics*, 8.

Mishra, D. and A. Sen (2012): "Roberts' Theorem with Neutrality: A Social Welfare Ordering Approach," *Games and Economic Behavior*, 75, 283–298.

Mitra, M. and A. Sen (2010): "Efficient Allocation of Heterogenous Commodities with Balanced Transfers," *Social Choice and Welfare*, 35, 29–48.

Moulin, H. (2009): "Almost Budget-Balanced VCG Mechanisms to Assign Multiple Objects," *Journal of Economic theory*, 144, 96–119.

Myerson, R. B. and M. A. Satterthwaite (1983): "Efficient Mechanisms for Bilateral trading," *Journal of economic theory*, 29, 265–281.

Nisan, N. and A. Ronen (2001): "Algorithmic Mechanism Design," *Games and Economic Behavior*, 35, 166–196.

Rahman, D. (2011): "Detecting Profitable Deviations," Working Paper, University of Minnesota.

Riley, J. G. (1988): "Ex Post Information in Auctions," *Review of Economic Studies*, 55, 409–429.

Roberts, K. (1979): *The Characterization of Implementable Choice Rules*, North Holland Publishing, chap. Aggregation and Revelation of Preferences, 321–348, editor: J-J. Laffont.

Rochet, J. C. (1987): "A Necessary and Sufficient Condition for Rationalizability in a Quasi-linear Context," *Journal of Mathematical Economics*, 16, 191–200.

Rockafellar, R. T. (1970): *Convex Analysis*, Princeton University Press.

Saks, M. E. and L. Yu (2005): "Weak Monotonicity Suffices for Truthfulness on Convex Domains," in *Proceedings of* 7^{th} *ACM Conference on Electronic Commerce*, ACM Press, 286–293.

Sen, A. K. (1970): *Collective Choice and Social Welfare*, vol. 5, Holden-Day.

Skrzypacz, A. (2013): "Auctions with Contingent Payments - An Overview," *International Journal of Industrial Organization*.

Suijs, J. (1996): "On Incentive Compatibility and Budget Balancedness in Public Decision Making," *Economic Design*, 2, 193–209.

Varian, H. R. (1982): "The Nonparametric Approach to Demand Analysis," *Econometrica*, 50, 945–973.

Vohra, R. V. (2011): *Mechanism Design: A Linear Programming Approach*, Cambridge University Press.