A Complexity Result for Undominated-Strategy Implementation

Gabriel Carroll, Stanford University

gdc@stanford.edu

December 20, 2014

Abstract

We prove that there exist social choice correspondences that can be implemented in undominated strategies, using finite mechanisms, but can require arbitrarily many strategies to do so, even when there are only two agents and two types of each agent. This is in sharp contrast with dominant-strategy implementation, where the revelation principle implies that only one strategy for each type of agent is needed; our result shows that no corresponding simplifying principle can exist for undominated strategies.

JEL Classifications: D82, C72, D71

Keywords: undominated strategies, implementation, revelation principle, complexity

1 Introduction

Mechanism design studies situations in which a designer sets up an interaction between agents — a *mechanism* — whose outcome will depend on the agents' privately known preferences in some desired way. An important modeling choice in any mechanism design problem is the choice of solution concept — that is, the assumptions the designer is willing to make about how agents will behave in any given mechanism.

One widely-used solution concept is *dominant-strategy* implementation. Here the designer constructs the mechanism so that each type of each agent has a dominant strategy

— a strategy that is optimal no matter what the other agents do — and so that a desirable outcome occurs as long as the agents play these strategies. This approach is about as robust as one could wish for: it makes no assumptions about agents' beliefs about each other's preferences or strategic behavior; as long as each agent knows his own preferences, he should be willing to play his dominant strategy, leading to the desired outcome. But because dominant-strategy implementation is so demanding, interesting positive results have been achieved only in some specific domains; in general, few things can be implemented in dominant strategies. (Classic impossibility results for dominant strategies include the Gibbard-Satterthwaite theorem [8, 12] on non-implementability of nontrivial voting rules, and the Green-Laffont theorem [9] on impossiblity of budget balance in a public projects environment, among many others. The survey by Barberá [2] gives an overview of many negative and positive results.)

An alternative approach is the *Bayesian* paradigm, where the designer assumes some specific prior belief over players' types, and (typically) assumes that the players all share this prior, when predicting behavior in a mechanism. This allows for some more positive results than dominant-strategy implementation; for example, attaining budget balance in public projects [1]. But the practical usefulness of this approach is limited because it requires the designer to accept strong assumptions about the agents' beliefs and strategic behavior in order to be confident that the mechanism will perform as planned. The "Wilson doctrine" [13] calls for analysis that does not depend on such strong assumptions.

A potentially appealing middle ground is implementation in *undominated strategies*: that is, seeking a mechanism that will ensure a desirable outcome as long as agents do not play (weakly) dominated strategies. This seems to share the robustness of dominant-strategy implementation, allowing the designer to be very agnostic about agents' beliefs about each other, while not being quite as restrictive. Several works exploring possibilities in some particular environments have suggested that this may be a fruitful way forward. Börgers [6] shows that one can get around the Gibbard-Satterthwaite impossibility theorem, implementing nontrivial outcomes in a voting environment. More recently, Yamashita [14] considers auction and bilateral trade environments, and shows examples where one can accomplish more (in terms of expected welfare) using undominated-strategy than dominant-strategy mechanisms, and also examples where one cannot. Interestingly, Yamashita shows that some of the analytical tools used to study dominant-strategy and Bayesian mechanism design problems in these environments partially carry over to study undominated strategies, raising the hope that the study of undominated-strategy implementation can become tractable.

One advantage of working in the dominant-strategy paradigm is that problems are simplified by the *revelation principle*. This principle says that to describe the outcomes that can be implemented by some mechanism, instead of considering the very large abstract space of all possible mechanisms, it suffices to focus on mechanisms where the strategies of each agent i are labeled by i's types, and each type plays the corresponding strategy. For example, if there are two agents, each with three types, it is sufficient for the analyst to restrict attention to mechanisms that have only three strategies for each agent. The same principle also applies to Bayesian mechanism design.[1]

By contrast, for implementation in undominated strategies, no such immediate reduction is available. If some desired outcome is implemented in undominated strategies by some mechanism, we cannot simplify the mechanism by choosing a particular strategy for each type and then throwing away all the unchosen strategies, as in dominant-strategy implementation. The problem is that when extra strategies for one player are thrown out, then strategies for another player that were originally dominated for some type might now become undominated, leading to unwanted outcomes. Thus, even with a small number of types, we might worry that undominated implementation could be possible only by offering a large number of strategies, with each strategy being needed to prevent some other strategy from becoming undominated. This suggests that studying undominated-strategy implementation might be much more complex than studying dominant-strategy (or Bayesian) implementation.

In this note, we prove a formal version of this complexity statement. We restrict ourselves to small environments: two agents, two types of each agent, and also private values (i.e. each agent knows his own preferences over outcomes). We show that, even in these small environments, there exist outcomes (SCC's) that can be implemented in undominated strategies, but that require arbitrarily many strategies in order to do so. We can think of this as a negative result for understanding undominated-strategy implementation: it shows that that no general-purpose simplifying principle, analogous to the revelation principle, can exist here.

We should make clear, however, that the result is not an insurmountable barrier to studying undominated-strategy implementation. Our result shows the absence of any general simplifying principle for arbitrary, unstructured environments. But in any particular structured environment — such as Yamashita's [14], with quasi-linear preferences

[1]With a caveat: it applies only to *partial* implementation, finding a mechanism with *some* good Bayesian Nash equilibrium; whereas the closest analogue to the question we study would consider *full* implementation, where *all* equilibria must ensure desirable outcomes.

and one-dimensional types — it may still be possible to give a clean characterization of what can and cannot be implemented.

There is not much prior literature giving general results for undominated-strategy implementation. Early work by Jackson [10] found that basically anything can be implemented in undominated strategies, but did so by creating infinite chains of strategies, each one dominated by the next, so that no strategy in the chain ends up undominated. This use of infinite chains is generally considered unpalatable. One natural restriction is to require the mechanism to be *bounded*, meaning that any dominated strategy should be dominated by some strategy that is not itself dominated. (In this note we will focus on finite mechanisms, which are necessarily bounded.) Jackson showed that any SCC implementable by such a mechanism satisfies a weak form of strategyproofness. In particular, a social choice *function* — specifying a single outcome for each type profile — can only be implemented in undominated strategies if it is implementable in dominant strategies, so that the revelation principle applies in this particular case.

One other prior paper with some similarity in spirit to ours is by Dutta and Sen [7] on *full* implementation in Bayesian equilibrium (see Footnote 1). They give a finite environment — two agents, two types of each, and two outcomes — and a social choice function that can be fully implemented, but only by an infinite mechanism. Their mechanism involves an infinite-chain construction to avoid unwanted equilibria. Another strand of work that similarly looks for a simplifying principle to reduce a space of mechanisms, with positive results, is that of Bester and Strausz [4, 3, 5] on mechanism design with limited commitment. Ours seems to be the first example of a negative result in this vein that focuses on finite but arbitrarily large mechanisms.

2 Model and result

Here and subsequently, we will use the notation $[k]$, with k a positive integer, for the set $\{1, 2, \ldots, k\}$. We will give definitions for general numbers of agents and types; later, we will specialize to consider just two agents and two types of each agent.

An *environment*, then, consists of the following primitives:

- a set $[n]$ of *agents*, for some n;
- for each agent i, a finite set of *types* T_i;
- a finite set $\mathcal{X}$ of possible *outcomes*;

- for each i and each $t_i \in T_i$, a *preference* $\succ_{t_i}$, which is a strict total ordering over $\mathcal{X}$.

Given such an environment, we write $T = T_1 \times \cdots \times T_n$ for the set of type profiles. A *social choice correspondence* (SCC) is a nonempty-valued correspondence $F : T \rightrightarrows \mathcal{X}$, specifying, for each profile of agents' types, the outcomes that the designer considers acceptable.

A *mechanism* consists of

- a nonempty, finite set S_i of *strategies* for each agent i, whose product is denoted by $S = S_1 \times \cdots \times S_n$; and
- an *outcome function* $g : S \to \mathcal{X}$.

Consider a given environment and a given mechanism. For each agent i we write $S_{-i} = \times_{j \neq i} S_j$. For a type t_i of agent i, say that strategy $s_i \in S_i$ is *dominated* by $s_i' \in S_i$ if, for all $s_{-i} \in S_{-i}$,

$$g(s_i', s_{-i}) \succeq_{t_i} g(s_i, s_{-i})$$

and there is some s_{-i} such that $g(s_i', s_{-i}) \neq g(s_i, s_{-i})$. A strategy s_i is *undominated* for t_i if there is no s_i' that dominates it.

We then say that the mechanism (S, g) *implements* the SCC F in undominated strategies if, for every profile of types $t = (t_1, \ldots, t_n) \in T$ and every strategy profile $s = (s_1, \ldots, s_n) \in S$ such that s_i is undominated for t_i (for each i), we have $g(s) \in F(t)$.

Note that this definition is what is sometimes called "weak implementation" (e.g. [11]): it requires only that every outcome of undominated strategies be consistent with F. Some of the relevant literature (including [10]) uses "strong implementation," which also requires that every outcome specified by F be attainable by some undominated strategies. Arguably weak implementation is the more natural desideratum for our designer to use, since she is agnostic about agents' behavior: even if F were strongly implemented, she could not guarantee that the agents would be willing to play so as to attain any particular element of $F(t)$. In any case, a version of our main result also holds with strong implementation; see below.

Note also that we here use weak dominance. This is consistent with recent literature [6, 14]. If we instead were to use strict dominance, then more strategies would be undominated in a given mechanism, so we might expect fewer things could be implemented. However, our theorem and proof would go through unchanged with strict dominance.

Here is our complexity result on undominated-strategy implementation.

Theorem 2.1. *For any positive integer M, there exists an environment with 2 agents, and 2 types of each agent, and an SCC F, such that*

- *F can be implemented in undominated strategies;*
- *any mechanism implementing F in undominated strategies must have at least M strategies for each agent.*

Note that the statement of this theorem immediately implies a corresponding statement for strong implementation. Indeed, if we consider the environment given by the theorem statement and the mechanism $((S_1, S_2), g)$, implementing it, then letting F' be the SCC strongly implemented by this mechanism (i.e. $F'(t)$ is the set of all values of $g(s_1, s_2)$, where s_i is undominated for t_i), it follows that F' cannot be (weakly or strongly) implemented with fewer than M strategies for each agent.

3 Patterns

At the heart of our result is a particular combinatorial construction. We will describe the construction abstractly here, and the proof of the main result will then illustrate how it relates back to undominated-strategy implementation.

Let $\mathcal{Y}$ be an arbitrary finite set, and let $\succ^1, \succ^2, \succ^3, \succ^4$ be four total orderings of $\mathcal{Y}$. If q, r are two positive integers, we will say that a function $\phi : [2q] \times [2r] \to \mathcal{Y}$ is a *(q, r)-pattern on* $(\mathcal{Y}, \succ^1, \succ^2, \succ^3, \succ^4)$ if all of the following conditions are satisfied:

(a) For each even $i \in [2q]$ and each $j \in [2r]$,

$$\phi(i, j) \succ^1 \phi(i-1, j);$$

(b) For each odd $i \in [2q]$ and each $j \in [2r]$,

$$\phi(i, j) \succ^2 \phi(i-1, j),$$

where the ordering is taken mod $2q$ (so we read $i - 1$ as $2q$ when $i = 1$);

(c) For each $i \in [2q]$ and each even $j \in [2r]$,

$$\phi(i, j) \succ^3 \phi(i, j-1);$$

(d) For each $i \in [2q]$ and each odd $j \in [2r]$,

$$\phi(i,j) \succ^4 \phi(i,j-1),$$

where the ordering is taken mod $2r$.

This concept is illustrated in Figure 1. A pattern is an assignment of elements of $\mathcal{Y}$ to the dots so that each solid arrow represents a preference according to $\succ^1$, with the element at the tail preferred over the element at the head; likewise, each dashed arrow represents a preference according to $\succ^2$, each dotted arrow $\succ^3$, and each dashed-dotted arow $\succ^4$.

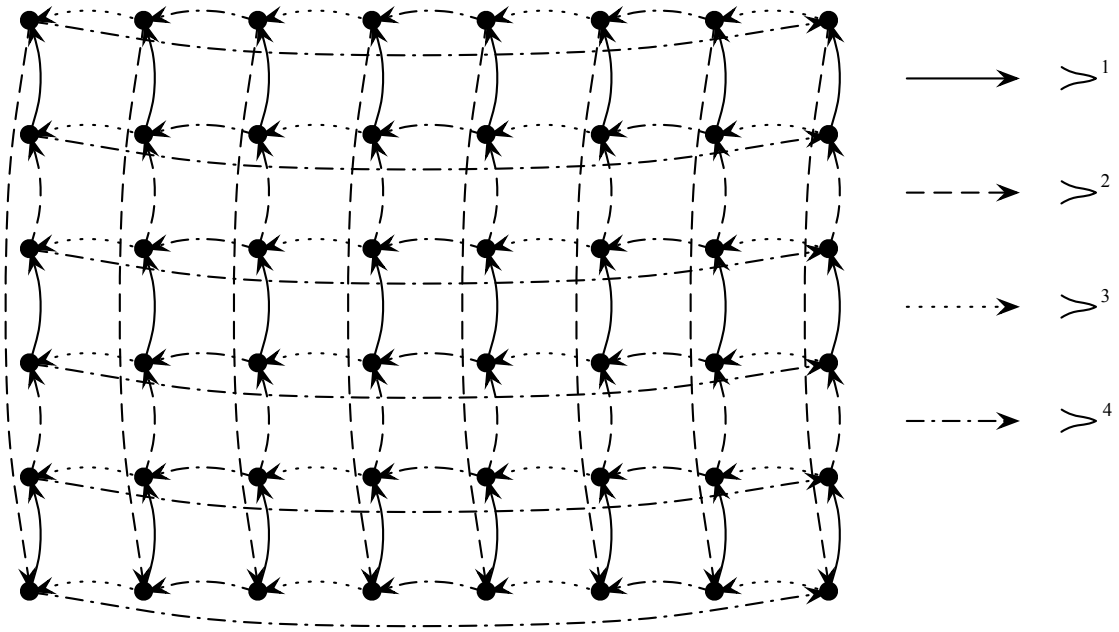

Figure 1: The concept of a pattern

The pattern is called *injective* if the function ϕ is injective.

Lemma 3.1. *For any positive integer M, it is possible to construct $\mathcal{Y}, \succ^1, \succ^2, \succ^3, \succ^4$ so that there exists an injective (M,M)-pattern, but there does not exist a (q,r)-pattern for any pair q,r with $\min\{q,r\} < M/2$.*

Proof: We will let $\mathcal{Y} = [2M] \times [2M]$. To describe the constructions of the orderings $\succ^1, \succ^2, \succ^3, \succ^4$, we will need some notation.

We first define functions $i^*, j^* : [2M] \to [2M]$ as follows:

- If j is even, then $i^*(j) = 2M + 2 - j$; otherwise, $i^*(j) = 2M$.
- If i is even and $i < 2M$, then $j^*(i) = 2M - i$; otherwise, $j^*(i) = 2M$.

Now, for any $k \in [2M]$, define a total ordering $\succ_-^{\langle k \rangle}$ on $[2M]$ as follows: for distinct i, i', we have $i \succ_-^{\langle k \rangle} i'$ iff

- $i > i' \geq k$, or
- $k > i > i'$, or
- $i' \geq k > i$.

Thus, numbers less than k are ordered above numbers greater than or equal to k, but within each of these two ranges, we follow the usual ordering. We also define a total ordering $\succ_+^{\langle k \rangle}$ to be the same as $\succ_-^{\langle k \rangle}$, except that k itself is ordered highest rather than lowest. Note that $\succ_+^{\langle k \rangle}$ is the same as $\succ_-^{\langle k+1 \rangle}$ (or, in case of $k = 2M$, we have $\succ_+^{\langle 2M \rangle} = \succ_-^{\langle 1 \rangle}$, both of which coincide with the usual ordering $>$). However we keep the redundant notation for convenience.

Now we define the four orderings as follows: For distinct pairs $(i, j), (i', j') \in [2M] \times [2M]$,

- $(i, j) \succ^1 (i', j')$ iff $j > j'$, or $j = j'$ and $i \succ_+^{\langle i^*(j) \rangle} i'$;
- $(i, j) \succ^2 (i', j')$ iff $j > j'$, or $j = j'$ and $i \succ_-^{\langle i^*(j) \rangle} i'$;
- $(i, j) \succ^3 (i', j')$ iff $i > i'$, or $i = i'$ and $j \succ_+^{\langle j^*(i) \rangle} j'$;
- $(i, j) \succ^4 (i', j')$ iff $i > i'$, or $i = i'$ and $j \succ_-^{\langle j^*(i) \rangle} j'$.

It is straightforward to see that each of these does indeed define a total ordering of $[2M] \times [2M] = \mathcal{Y}$. For example, $\succ^1$ above defines a lexicographic comparison, where two pairs are compared first by their second coordinate, and if the second coordinates coincide, then they are compared by the first coordinate (using an ordering that depends on what the second coordinate is). Similarly for $\succ^2, \succ^3, \succ^4$.

We check the existence of an injective $(2M, 2M)$-pattern ϕ^*. We claim we can simply define $\phi^* : [2M] \times [2M] \to \mathcal{Y}$ to be the identity, $\phi^*(i, j) = (i, j)$.

Let us check that the requirements are met. To see that (a) is satisfied, say, we need to know that $i \succ_+^{\langle i^*(j) \rangle} i - 1$ for all even i and all j. Since $i^*(j)$ is always even, the numbers $i, i - 1$ are either both $\leq i^*(j)$ or both $> i^*(j)$, so $\succ_+^{\langle i^*(j) \rangle}$ orders them in the same way

as $>$, which is what we need. For (b), we need that $i \succ_-^{\langle i^*(j)\rangle} i-1$ for all odd i and all j. Since $i^*(j)$ is even, we have either $i, i-1$ both $\geq i^*(j)$ or both $< i^*(j)$. The only extra hitch is the case $i=1$ and $i-1=2M$, in which case we again have $i \succ_-^{\langle i^*(j)\rangle} i-1$ (noticing that $i^*(j) \neq 1$). Checking conditions (c) and (d) is totally analogous, using the fact that $j^*(i)$ is even for all i.

Now for the final assertion of the lemma, suppose that a (q,r)-pattern exists for some q and r. Call it ϕ. We will show that $q \geq M/2$ and $r \geq M/2$. We will write $\phi_1(i,j)$ for the first component of $\phi(i,j)$, and write $\phi_2(i,j)$ similarly.

First, fix any j. We have the cycle

$$\phi(2q,j) \succ^1 \phi(2q-1,j) \succ^2 \phi(2q-2,j) \succ^1 \phi(2q-3,j) \succ^2 \cdots \succ^1 \phi(1,j) \succ^2 \phi(2q,j).$$

The second components, $\phi_2(i,j)$, can never increase along this cycle. Therefore, they must be constant: For each $j \in [2q]$, there is some number $\widehat{j}(j)$ such that $\phi_2(i,j) = \widehat{j}(j)$ for all i. Moreover, there must exist some i such that $\phi_1(i,j) = i^*(\widehat{j}(j))$. Here is why; for brevity we write $i^* = i^*(\widehat{j}(j))$. Since $\succ_+^{\langle i^*\rangle}$ and $\succ_-^{\langle i^*\rangle}$ agree on the ordering of the set $[2M] \setminus \{i^*\}$, if all $\phi(i,j)$ had first component different from i^*, then $\succ^1$ and $\succ^2$ would order all such pairs in the same way, contradicting the existence of the cycle. To avoid such a contradiction, we must have some pair with first component i^* appearing in the cycle.

So, we can define functions $\widehat{j} : [2r] \to [2M]$ and $\widetilde{i} : [2r] \to [2q]$ such that $\phi_2(i,j) = \widehat{j}(j)$ for all i,j, and $\phi_1(\widetilde{i}(j),j) = i^*(\widehat{j}(j))$ for all j.

Reversing the roles of i,j lets us define functions $\widehat{i}$ and $\widetilde{j}$ similarly. Summing up, we have defined functions $\widehat{i} : [2q] \to [2M]$, $\widehat{j} : [2r] \to [2M]$, and $\widetilde{i} : [2r] \to [2q]$, $\widetilde{j} : [2q] \to [2r]$, such that

$$\phi(i,j) = (\widehat{i}(i), \widehat{j}(j)) \qquad \text{for all } (i,j),$$

and

$$\widehat{i}(\widetilde{i}(j)) = i^*(\widehat{j}(j)); \qquad \widehat{j}(\widetilde{j}(i)) = j^*(\widehat{i}(i)).$$

Now define a sequence $i_1, i_2, \ldots$ by $i_1 = 1$ and $i_{k+1} = \widetilde{i}(\widetilde{j}(i_k))$. We can define corresponding values $\widehat{i}_1, \widehat{i}_2, \ldots$ by $\widehat{i}_k = \widehat{i}(i_k)$. These values satisfy the recursion $\widehat{i}_{k+1} = i^*(j^*(\widehat{i}_k))$, since

$$\widehat{i}_{k+1} = \widehat{i}(\widetilde{i}(\widetilde{j}(i_k))) = i^*(\widehat{j}(\widetilde{j}(i_k))) = i^*(j^*(\widehat{i}(i_k))).$$

From the definitions of i^* and j^*, we can see that

- if $\widehat{i}_k$ is even and $\widehat{i}_k < 2M$, then $\widehat{i}_{k+1} = \widehat{i}_k + 2$;

- otherwise, $\widehat{i}_{k+1} = 2$.

So the sequence of values $(\widehat{i}_k)$ enters the cycle $2, 4, 6, 8, \ldots, 2M-2, 2M, 2, 4, 6, \ldots$. In particular, there are at least M distinct values. But each $\widehat{i}_k$ equals the value of the function $\widehat{i}$ on some $i \in [2q]$. Therefore, we must have $2q \geq M$ or $q \geq M/2$.

A similar argument shows that $r \geq M/2$. □

4 The main proof

Now we can show how the construction from the previous section is used to prove the main theorem.

Proof of Theorem 2.1: Let $\mathcal{Y}, \succ^1, \succ^2, \succ^3, \succ^4$, and the injective pattern ϕ^* be as given in the lemma. We take the environment to consist of two types of agent 1, $\{t_1^E, t_1^O\}$, and two types of agent 2, $\{t_2^E, t_2^O\}$, and the set of outcomes $\mathcal{X} = \mathcal{Y}$. The preferences are as follows: t_1^E has preference $\succ^1$; t_1^O has preference $\succ^2$; t_2^E has preference $\succ^3$; and t_2^O has preference $\succ^4$.

The social choice correspondence F is defined by

$$F(\{t_1^E, t_2^E\}) = \{\phi^*(i,j) \mid i \text{ even}, j \text{ even}\};$$

$$F(\{t_1^E, t_2^O\}) = \{\phi^*(i,j) \mid i \text{ even}, j \text{ odd}\};$$

$$F(\{t_1^O, t_2^E\}) = \{\phi^*(i,j) \mid i \text{ odd}, j \text{ even}\};$$

$$F(\{t_2^O, t_2^E\}) = \{\phi^*(i,j) \mid i \text{ odd}, j \text{ odd}\}.$$

First we check that F can be implemented in undominated strategies. Let $S_i = [2M]$ for each $i = 1, 2$, and $g(i,j) = \phi^*(i,j)$. This defines the mechanism.

For type t_1^E of agent 1, the strategy i dominates $i-1$ for any even $i \in [2M]$. Indeed, by construction, $\phi^*(i,j) \succ^1 \phi^*(i-1,j)$ for every j, which is exactly the same as saying that t_1^E prefers $g(i,j)$ over $g(i-1,j)$. Similarly, for type t_1^O, strategy i dominates $i-1$ for any odd i (where we read $i-1$ as $2M$ for $i=1$). And likewise, for type t_2^E, i dominates $i-1$ whenever i is even; for type t_2^O, i dominates $i-1$ whenever i is odd.

Now, suppose each agent i is of type t_i and plays a strategy s_i that is undominated for that type.

If the realized type profile is (t_1^E, t_2^E), then s_1, s_2 must both be even, so the outcome $g(s_1, s_2) = \phi^*(s_1, s_2)$ is in $F(t_1^E, t_2^E)$. The cases of the other three type profiles — (t_1^E, t_2^O),

(t_1^O, t_2^E), (t_1^O, t_2^O) — are similar. In each case, we have $g(s_1, s_2) \in F(t_1, t_2)$. Thus, the proposed mechanism does indeed implement F.

Now suppose $((S_1, S_2), g)$ is any mechanism implementing F in undominated strategies. Let S_1^E be the set of strategies that are undominated for t_1^E, and S_1^O, S_2^E, S_2^O similarly. Each of these is nonempty — since the mechanism is finite, every type has some undominated strategy. Note that $S_1^E \cap S_1^O = \emptyset$: otherwise, if s_1 lies in both, then we can take s_2 to be any element of S_2^E (say), and then $g(s_1, s_2)$ needs to lie in both $F(t_1^E, t_2^E)$ and $F(t_1^O, t_2^E)$; but these two sets are disjoint (since ϕ^* is injective). Thus, any $s \in S_1^E$ is dominated by some s' for type t_1^O. We may assume $s' \in S_1^O$ (if not, s' is itself dominated for t_1^O by another strategy in S_1^O, so we take that strategy in place of s'). Likewise, any $s \in S_1^O$ is dominated for type t_1^E by some $s' \in S_1^E$. Following these relations gives us a sequence of strategies, alternating between members of S_1^E and S_1^O, such that each is dominated by the next one for t_1^O (respectively, t_1^E). But since $S_1^E \cup S_1^O$ is finite, these must eventually form a cycle. Thus, after reindexing, we have a cycle of $2q$ distinct strategies $s_1^1, s_1^2, s_1^3, \ldots, s_1^{2q}$, for some positive integer q, such that

- if i is even, $s_1^i \in S_1^E$, and s_1^i dominates s_1^{i-1} for type t_1^E;
- if i is odd, $s_1^i \in S_1^O$, and s_1^i dominates s_1^{i-1} for type t_1^O (as usual, indexing in the cycle is mod $2q$).

Likewise for agent 2: S_2^E and S_2^O are disjoint, and we find a cycle of $2r$ distinct strategies $s_2^1, s_2^2, \ldots, s_2^{2r}$, for some r, such that

- if i is even, $s_2^i \in S_2^E$, and s_2^i dominates s_2^{i-1} for type t_2^E;
- if i is odd, $s_2^i \in S_2^O$, and s_2^i dominates s_2^{i-1} for type t_2^O (indexing mod $2r$).

Then, defining $\phi : [2q] \times [2r] \to \mathcal{Y}$ by $\phi(i, j) = g(s_1^i, s_2^j)$ gives a $(2q, 2r)$-pattern. To see this, notice, for example, that for all j and all even i, the statement that s_1^i dominates s_1^{i-1} for type t_1^E — whose preference is $\succ^1$ — implies that $g(s_1^i, s_2^j) \succeq^1 g(s_1^{i-1}, s_2^j)$. We cannot have equality, because $g(s_1^i, s_2^j) \in F(t_1^E, t_2^E)$ or $F(t_1^E, t_2^O)$ (depending on the parity of j) and $g(s_1^{i-1}, s_2^j) \in F(t_1^O, t_2^E)$ or $F(t_1^O, t_2^O)$, and in either case $F(t_1^E, t_2)$ is disjoint from $F(t_1^O, t_2)$. Thus, we have $g(s_1^i, s_2^j) \succ^1 g(s_1^{i-1}, s_2^j)$, for each even i and all j. This is statement (a) of the definition of a pattern. Similar reasoning checks that (b), (c), and (d) are also satisfied.

Then, by the lemma, we have $q, r \geq M/2$. Since our cycles consist of $2q$ distinct strategies in S_1 and $2r$ distinct strategies in S_2, we see that each agent's strategy set contains at least M strategies. □

5 Minimality

There are two respects in which one might potentially seek a construction that is simpler than the one we have given. First, we have two agents with two types each; is it possible to give a construction where only one agent has more than one type? And second, our F becomes arbitrarily large as M grows; is it possible to give examples of SCC's with only a fixed number of acceptable outcomes, yet requiring unbounded numbers of strategies to implement? The answers to both questions are no. This should not be surprising, but it still requires checking.

Here is our answer to the first question:

Proposition 5.1. *Consider an environment in which only one agent has more than one type. If F is an SCC that can be implemented in undominated strategies, then it can be so implemented by a mechanism (S', g') with $S'_i = T_i$ for each agent i.*

The proof is simple, but does require a little care. We can't simply choose one strategy for each one-type agent and throw away the other strategies, as in the usual revelation principle: doing so might cause formerly dominated strategies for the multiple-type agent to become undominated. (If we used strict rather than weak dominance, this would not be the case.)

Proof: Without loss of generality, assume agent 1 is the only one who may have multiple types. Let (S, g) be a mechanism implementing F. Define (S', g') as follows. For each agent i, $S'_i = T_i$. For each type profile $t = (t_1, \ldots, t_n)$, consider all the outcomes $g(s_1, \ldots, s_n)$ that can be obtained from the original mechanism, where each s_i is undominated for t_i; among all such outcomes, define $g'(t)$ be the one that is most preferred by t_1.

Evidently $g'(t) \in F(t)$, since g implemented F. So we just have to check that play of undominated strategies at t necessarily leads to outcome $g'(t)$. Since every agent other than agent 1 only has one choice of strategy, this is equivalent to checking that the best obtainable outcome, according to the preference of t_1, is $g'(t)$.

Suppose not, so that there is some outcome $x \succ_{t_1} g'(t)$ obtainable under g'. Hence, $x = g(s_1, \ldots, s_n)$, where s_1 is some strategy in S_1, and for each $i > 1$, $s_i \in S_i$ is some undominated strategy for (the unique type of) agent i. Then, either s_1 is dominated for t_1, in which case we can find some strategy s'_1 dominating it that is not itself dominated; or s_1 is undominated for t_1, in which case we take $s'_1 = s_1$. In either case, put $x' = g(s'_1, s_2, \ldots, s_n)$, and the dominance relation implies $x' \succeq_{t_1} x$. But by construction of g' we have $g'(t) \succeq_{t_1} x'$. Hence, $g'(t) \succeq_{t_1} x$, contradicting our assumption. □

And for the second question, we show that in order to construct examples requiring unboundedly many strategies, we do indeed need unboundedly many outcomes, at least if we keep the number of agents small. The proof is straightforward: if there is a fixed number k of outcomes, then there are effectively at most $k!$ possible types of each agent, so there are really only finitely many SCC's to consider, and by allowing a large enough number of strategies we can implement any of them.

Proposition 5.2. *Let n, k be positive integers. Then, there exists M with the following property: For any environment with n agents, and any SCC F whose range contains no more than k outcomes, if F can be implemented in undominated strategies, then it can be so implemented by a mechanism that has at most M strategies for each agent.*

Proof: Let $\widehat{\mathcal{X}}$ be the range of F. Define a new environment as follows: The agents $i = 1, \ldots, n$ are the same as in the original environment. The set of outcomes is $\widehat{\mathcal{X}}$. For each agent i, the set of types $\widehat{T}_i$ is simply the set of all preference orderings on $\widehat{\mathcal{X}}$ that are induced by some type in T_i; for each such type $\succ_i$, we define its preference on $\widehat{\mathcal{X}}$ to be simply the preference $\succ_i$ itself. Although types and their preferences are mathematically the same object in this environment, they play different roles in the model, so we will use the notation $\widehat{t}_i$ to refer to a type in $\widehat{T}_i$, and $\succ_{\widehat{t}_i}$ to refer to the corresponding preference. For each $\widehat{t}_i \in \widehat{T}_i$, let $T_i(\widehat{t}_i)$ in T_i be the (nonempty) set of original types that have the corresponding induced preference over $\widehat{\mathcal{X}}$. For a profile $\widehat{t} \in \widehat{T}$, let $T(t) = \times_{i=1}^{n} T_i(\widehat{t}_i)$.

Define the SCC $\widehat{F} : \widehat{T} \rightrightarrows \widehat{\mathcal{X}}$ as follows: for each $\widehat{t} = (\widehat{t}_1, \ldots, \widehat{t}_n) \in \widehat{T}$, let $\widehat{F}(\widehat{t}) = \cap_{t \in T(\widehat{t})} F(t)$. We will show that $\widehat{F}$ can be implemented, in the new environment. (In particular, this will verify that $\widehat{F}$ is nonempty-valued.)

Suppose the original SCC F was implemented in the original environment by some mechanism (S, g). We cannot immediately reinterpret (S, g) as a mechanism for the new environment, because g may specify outcomes that no longer exist in the new environment — but only as a result of dominated strategies. Thus, for each i, let $\widehat{S}_i \subseteq S_i$ be the set of all strategies that are undominated for some type. Whenever s is a strategy profile with $s_i \in \widehat{S}_i$ for each i, then $g(s) \in F(t)$ for some type profile t, and so $g(s) \in \widehat{\mathcal{X}}$. Thus, we have a mechanism over the new environment, defined by taking each agent i's strategy set to be $\widehat{S}_i$, and defining $\widehat{g} : \widehat{S} \to \widehat{\mathcal{X}}$ as the restriction of g.

We claim that this new mechanism implements $\widehat{F}$ over $\widehat{T}$. Indeed: Let $\widehat{t}_1, \ldots, \widehat{t}_n$ be types in $\widehat{\mathcal{X}}$, and let $\widehat{s}_1, \ldots, \widehat{s}_n$ be corresponding undominated strategies. Let $t_i \in T_i(\widehat{t}_i)$ for each i. We need to show that $\widehat{g}(\widehat{s})$ is in $F(t)$. For each i, either $\widehat{s}_i$ was undominated for t_i in the original mechanism, or it was dominated by some other strategy $\widehat{s}'_i$ that is in turn

undominated for t_i. In the former case we take $\widehat{s}'_i = \widehat{s}_i$. So in either case, $\widehat{s}'_i \in \widehat{S}_i$. We claim that $\widehat{s}'_i$ is equivalent to $\widehat{s}_i$ in $\widehat{S}$ — that is, for any $\widehat{s}''_{-i} \in \widehat{S}_{-i}$, we have $\widehat{g}(\widehat{s}'_i, \widehat{s}''_{-i}) = \widehat{g}(\widehat{s}_i, \widehat{s}''_{-i})$. If $\widehat{s}'_i = \widehat{s}_i$ this is obvious; otherwise, $\widehat{s}'_i$ weakly dominates $\widehat{s}_i$ for t_i over S, hence

$$\widehat{g}(\widehat{s}'_i, \widehat{s}''_{-i}) \succeq_{t_i} \widehat{g}(\widehat{s}_i, \widehat{s}''_{-i}).$$

Since both $\widehat{g}(\widehat{s}'_i, \widehat{s}''_{-i})$ and $\widehat{g}(\widehat{s}_i, \widehat{s}''_{-i})$ are in $\widehat{\mathcal{X}}$, we can actually write this as

$$\widehat{g}(\widehat{s}'_i, \widehat{s}''_{-i}) \succeq_{\widehat{t}_i} \widehat{g}(\widehat{s}_i, \widehat{s}''_{-i}).$$

But since $\widehat{s}_i$ is undominated for $\widehat{t}_i$, this is possible only if there is equality, $\widehat{g}(\widehat{s}'_i, \widehat{s}''_{-i}) = \widehat{g}(\widehat{s}_i, \widehat{s}''_{-i})$, for all $\widehat{s}''_{-i}$, as claimed. Now, the equivalence just shown, applied to every agent i in succession, gives us that

$$\widehat{g}(\widehat{s}_1, \ldots, \widehat{s}_n) = \widehat{g}(\widehat{s}'_1, \ldots, \widehat{s}'_n) = g(\widehat{s}'_1, \ldots, \widehat{s}'_n) \in F(t_1, \ldots, t_n)$$

which is what we needed. Thus, the new mechanism does indeed implement $\widehat{F}$.

Now, our reduced environment consists of n agents, at most k outcomes, and at most $k!$ types of each agent. There are only finitely many SCC's for such an environment (up to isomorphism), and so there exists some M, depending only on n and k, such that any such SCC that is implementable in undominated strategies can be implemented with a mechanism of at most M strategies for each agent. Consequently, for this M, there is some mechanism that implements our $\widehat{F}$ over $\widehat{T}$, using at most M strategies for each agent. Call it $(\widetilde{S}, \widetilde{g})$. We wrap up the proof by showing that such a mechanism implements F over T as well.

Let $t \in T$, and let $\widetilde{s}$ be a profile of corresponding undominated strategies. For each i, let $\widehat{t}_i \in \widehat{T}_i$ be the type whose preference over $\widehat{\mathcal{X}}$ agrees with t_i. Then $\widetilde{s}_i$ is also undominated for $\widehat{t}_i$, since all possible outcomes of the mechanism $(\widetilde{S}, \widetilde{g})$ are in the set $\widehat{\mathcal{X}}$. Hence,

$$\widetilde{g}(\widetilde{s}) \in \widehat{F}(\widehat{t}) \subseteq F(t).$$

Hence the mechanism implements F. □

References

[1] Claude d'Aspremont and Louis-André Gérard-Varet (1979), "Incentives and incomplete information," *Journal of Public Economics* 11 (1): 25–45.

[2] Salvador Barberá (2001), "An introduction to strategy-proof social choice functions," *Social Choice and Welfare* 18 (4): 619–653.

[3] Helmut Bester and Roland Strausz (2000), "Imperfect commitment and the revelation principle: the multi-agent case," *Economics Letters* 69 (2): 165–171.

[4] Helmut Bester and Roland Strausz (2001), "Contracting with imperfect commitment and the revelation principle: the single agent case," *Econometrica* 69 (4): 1077–1098.

[5] Helmut Bester and Roland Strausz (2007), "Contracting with imperfect commitment and noisy communication," *Journal of Economic Theory* 136 (1): 236–259.

[6] Tilman Börgers (1991), "Undominated strategies and coordination in normalform games," *Social Choice and Welfare* 8 (1): 65–78.

[7] Bhaskar Dutta and Arunava Sen (1994), "Bayesian implementation: The necessity of infinite mechanisms," *Journal of Economic Theory* 64 (1): 130–141.

[8] Gibbard, A. (1973): "Manipulation of voting schemes: a general result," *Econometrica*, 41, 587-601.

[9] Green, J. R., and J.-J. Laffont (1979): *Incentives in public decision-making.* Amsterdam: North-Holland.

[10] Matthew O. Jackson (1992), "Implementation in undominated strategies: a look at bounded mechanisms," *Review of Economic Studies* 59 (4): 757–775.

[11] Matthew O. Jackson (2001), "A crash course in implementation theory," *Social Choice and Welfare* 18 (4): 655–708.

[12] Satterthwaite, M. A. (1975): "Strategy-proofness and Arrow's conditions: existence and correspondence theorems for voting procedures and social welfare functions," *Journal of Economic Theory*, 10, 187–217.

[13] Robert Wilson (1987), "Game theoretic analysis of trading processes," in *Advances in economic theory: fifth world congress*, ed. Truman F. Bewley, Cambridge: Cambridge University Press: 33–70.

[14] Takuro Yamashita (2014), "Implementation in weakly undominated strategies, with applications to auctions and bilateral trade," *Review of Economic Studies*, forthcoming.