SUFFICIENCY OF WEAK MONOTONICITY FOR IMPLEMENTABILITY IN MAX DOMAIN *

Satish Agarwal[1] and Souvik Roy[2]

[1]Total SA, Reservoir Engineering Department, La defence, France

[2]Economic Research Unit, Indian Statistical Institute, Kolkata

Abstract

We consider implementation of a deterministic allocation rule using transfers in quasi-linear private values environments. We show that in a domain where the valuation of a set of objects is given by the maximum valuation in that set, an allocation rule is implementable if and only if it satisfies a familiar and simple condition called weak monotonicity or 2-cycle monotonicity.

1 Introduction

A classical problem in mechanism design is to investigate conditions which are necessary and sufficient for implementing an allocation rule. We investigate this question in private value and quasi-linear utility environment when the set of alternatives is finite and the allocation rule is deterministic (i.e not random). An allocation rule in such an environment is implementable if there exists a payment rule such that truth telling is a dominant strategy for the agents in the resulting mechanism. Our main result is that in a Max Domain, weak monotonicity implies K-cycle monotonicity. The weak monotonicity condition requires the following: given the types of other agents, if the alternative chosen by the allocation rule is a when agent i reports its type to be t and the alternative chosen by the allocation rule is b when agent i reports its type to be s, then it must be that

$$t(a) - t(b) \geq s(a) - s(b).$$

One of the earliest papers to identify the necessary and sufficient conditions for implementability was Rochet (1987), who proved that a condition called cycle monotonicity was needed for implementability in any type space. Myerson (1981) formally establishes that in a single object auction set up, where the type is single dimensional, weak monotonicity is necessary and sufficient condition for implementation. When the type space is multidimensional, if the set of alternatives is finite and the type space is convex, weak monotonicity implies cycle monotonicity (Bikhchandani et al. (2006), Saks and Yu (2005), Ashlagi et al. (2010)). Though convexity is a natural geometric

*The authors would like to gratefully acknowledge Debasis Mishra for his insightful comments.

property satisfied in many economic environments, it excludes many interesting type spaces. Recently, it has been shown that in Single Peaked Type Spaces, which is a non-convex domain, 2-cycle monotonicity implies higher cycle monotonicity (Mishra et al. (2014)). Moreover, the dichotomous domain is the only familiar domain known in literature where weak monotonicity is not sufficient for implementability but 2-cycle and 3-cycle monotonicity together are sufficient (Mishra and Roy (2013)). Indeed, our Max Domain type space is non-convex. To our knowledge, this paper is the first to examine the monotonicity of this specific domain. A characterization of implementability (given by cycle monotonicity) using weak monotonicity is useful because the cycle monotonicity is a difficult condition to use and interpret. On the other hand, weak monotonicity is a simpler condition. For this reason, 2-cycle monotonicity is often referred to as weak monotonicity (Bikhchandani et al. (2006)) or monotonicity (Ashlagi et al. (2010)). The next step to show is that weak monotonicity implies cycle monotonicity for a domain where the equality conditions of the max domain are replaced by inequalities, thus giving us Gross Substitution Domain.

2 Model

We consider a set N of n players and a set $S = \{a_1, a_2, \ldots, a_m\}$ of m objects. The set of alternatives $\mathcal{A}$ is defined as the set of all subsets of S.

Definition 2.1 (Max Domain)**.** Max domain $\mathcal{D}$ over a set S of objects is defined as $\mathcal{D} = \{t \in \mathbb{R}^{2^m} : t(X) = \max_{a \in X} t(\{a\}) \ \forall X \in \mathcal{A}\}$.

Notation. We introduce the following notation: $\tau(X, t_i) = \{x \in X : t_i(x) \geq t_i(y) \text{ for all } y \in X\}$ where $X \subseteq S$.

Definition 2.2. An allocation rule f is a function from $\mathcal{D}^n$ to $\mathcal{A}$.

Definition 2.3. An allocation rule f satisfies k-cycle monotonicity if $\forall i \in N, \forall t_1, t_2, \ldots, t_k \in \mathcal{D}$ and $\forall t_{-i} \in \mathcal{D}^{n-1}$,

$$\sum_{j=1}^{k} t_j(f(t_j, t_{-i})) - t_j(f(t_{j+1}, t_{-i})) \geq 0$$

where $t_{k+1} = t_1$.

Definition 2.4. An allocation rule f is implementable if there exists a payment function $p : \mathcal{D}^n \to \mathbb{R}$ such that $\forall i \in N, \forall t_i, t_i' \in \mathcal{D}$, and $\forall t_{-i} \in \mathcal{D}^{n-1}$

$$t_i\left(f(t_i, t_{-i})\right) - p(t_i, t_{-i}) \geq t_i\left(f(t_i', t_{-i})\right) - p(t_i', t_{-i}).$$

Definition 2.5. An allocation rule is said to satisfy weak monotonicity if it satisfies 2-cycle monotonicity.

3 Results

In this section we present the main result of this paper. First we present a theorem due to Rockafellar (1970) that states that k-cycle monotonicity for all $k \in \mathbb{N}$ is sufficient for implementability.

Theorem 3.1 (Rockafellar (1970)). *An allocation rule f is implementable if and only if it is k-cycle monotone for all $k \in \mathbb{N}$.*

We now present the main theorem of this paper. We show that the max domain is weak monotone domain, meaning that every rule that satisfies weak monotonicity must satisfy k-cycle monotonicity for all $k \in \mathbb{N}$ and hence will be implementable.

Theorem 3.2. *Weak monotonicity is sufficient for implementability in max domain.*

The proof of the theorem follows from the following lemma.

Lemma 3.1. *If an allocation rule f defined on max domain satisfies $(k-1)$-cycle monotonicity then it satisfies k-cycle monotonicity.*

Proof. Suppose not. Then there exists an allocation rule f and k type profiles $(t_j, t_{-i}); j = 1, \dots k$ consisting of k types $t_1, t_2, t_3, \dots, t_k \in \mathcal{D}$ of a player i and a type profile $t_{-i} \in \mathcal{D}^{n-1}$ of all players except i such that f satisfies $(k-1)$-cycle monotonicity for all size $(k-1)$ subsets of those k type profiles but violates k-cycle monotonicity over those k type profiles. Suppose $f(t_j, t_{-i}) = A_j$ for $j = 1, \dots, k$. We assume $t_l(A_1) = \alpha \ \forall l = 1, 2, 3, \dots, k$. This is without loss of generality as cycle monotonicity condition is location invariant. Applying $(k-1)$-cycle monotonicity over types $t_1, t_2, t_3, \dots, t_{k-1}$ and using assumption $t_1(A_1) = t_{k-1}(A_1)$ we have

$$t_2(A_2) + t_3(A_3) + \dots + t_{k-1}(A_{k-1}) \geq t_1(A_2) + t_2(A_3) + \dots + t_{k-2}(A_{k-1}) \tag{1}$$

Applying $(k-1)$-cycle monotonicity over types $t_2, t_3, t_4, \dots, t_k$ we have

$$t_2(A_2) + t_3(A_3) + \dots + t_k(A_k) \geq t_2(A_3) + \dots + t_{k-1}(A_k) + t_k(A_2) \tag{2}$$

Since f violates k-cycle monotonicity over $t_1, t_2, t_3, \dots, t_k$ and $t_1(A_1) = t_k(A_1)$ by our assumption, we have

$$t_2(A_2) + t_3(A_3) + \dots + t_k(A_k) < t_1(A_2) + t_2(A_3) + \dots + t_{k-1}(A_k) \tag{3}$$

Note that if

$$t_k(A_k) \geq t_{k-1}(A_k)$$

then (1) contradicts (3). Moreover, if

$$t_k(A_2) \geq t_1(A_2)$$

then (2) contradicts (3).

Suppose

$$\tau(A_2, t_1) = \tau(A_k, t_{k-1}).$$

Since this means $\tau(A_2, t_1) \in A_k$, we have $t_1(A_k) \geq t_1(A_2)$. Similarly we have $t_{k-1}(A_2) \geq t_{k-1}(A_k)$. Now 2-cycle monotonicity over types t_1 and t_k gives $t_k(A_k) \geq t_1(A_k)$. This along with $t_1(A_k) \geq t_1(A_2)$ implies $t_k(A_k) \geq t_1(A_2)$. Applying $(k-2)$-cycle monotonicity over types $t_2, \dots, t_{k-1}$ we have

$$t_2(A_2) + t_3(A_3) + \dots + t_{k-1}(A_{k-1}) \geq t_2(A_3) + \dots + t_{k-2}(A_{k-1}) + t_{k-1}(A_2).$$

Using $t_{k-1}(A_2) \geq t_{k-1}(A_k)$ and $t_k(A_k) \geq t_1(A_2)$ in the above equation we have

$$t_2(A_2) + t_3(A_3) + \ldots + t_{k-1}(A_{k-1}) + t_k(A_k) \geq t_1(A_2) + t_2(A_3) + \ldots + t_{k-1}(A_k)$$

which violates (3).

Now, we consider the following remaining case

$$t_{k-1}(A_k) > t_k(A_k), \tag{4}$$

$$t_1(A_2) > t_k(A_2), \tag{5}$$

and

$$\tau(A_2, t_1) \neq \tau(A_k, t_{k-1}). \tag{6}$$

Take $b, c \in S$ such that $b \neq c$, $b \in \tau(A_2, t_1)$ and $c \in \tau(A_k, t_{k-1})$. Note that this is possible by assumption (6). Consider a type t_{k+1} such that

$$\begin{aligned} t_{k+1}(\{b\}) &= t_1(\{b\}) - \epsilon, \\ t_{k+1}(\{c\}) &= t_k(A_k) + \epsilon, \text{ and} \\ t_{k+1}(\{x\}) &= t_k(\{x\}) \textit{ for all } x \in S \setminus \{b, c\} \end{aligned}$$

where $\epsilon > 0$ is arbitrarily small.

Claim 3.1. *For the type t_{k+1}*

$$t_{k+1}(X) \geq t_k(X) \textit{ for all } X \in \mathcal{A}. \tag{7}$$

Proof. By the construction of t_{k+1} it is enough to show that $t_{k+1}(\{b\}) > t_k(\{b\})$ and $t_{k+1}(\{c\}) > t_k(\{c\})$. The fact that $t_{k+1}(\{c\}) > t_k(\{c\})$ follows from the fact that $c \in A_k$, as then $t_k(A_k) \geq t_k(\{c\})$ and hence $t_{k+1}(\{c\}) = t_k(A_k) + \epsilon > t_k(\{c\})$. We now proceed to show that $t_{k+1}(\{b\}) > t_k(\{b\})$. Since $b \in \tau(A_2, t_1)$, $t_1(\{b\}) = t_1(A_2)$. Hence by (5) $t_1(\{b\}) > t_k(A_2) \geq t_k(\{b\})$. Here the last inequality follows from the fact that $b \in A_2$. Since ϵ is arbitrarily small, $t_{k+1}(\{b\}) = t_1(\{b\}) - \epsilon > t_k(\{b\})$, which completes the proof of the Claim 3.1.

Claim 3.2. *It must be that $t_{k+1}(A_1) = \alpha$.*

Proof. Note that by the construction of t_{k+1}, $t_{k+1}(A_1) = t_k(A_1) = \alpha$ if $b, c \notin A_1$. We first show that $b \notin \tau(A_1, t_{k+1})$. If $b \notin A_1$ then there is nothing to prove. Suppose $b \in A_1$ which means $t_1(A_1) \geq t_1(\{b\})$ or, $t_1(A_1) > t_1(\{b\}) - \epsilon = t_{k+1}(\{b\})$. Since $t_1(A_1) = t_k(A_1) = \alpha$, Claim 3.1 implies $t_{k+1}(A_1) \geq t_k(A_1) > t_{k+1}(\{b\})$. This proves that $b \notin \tau(A_1, t_{k+1})$.

Now we show $c \notin \tau(A_1, t_{k+1})$. As before if $c \notin A_1$ then there is nothing to prove. Suppose $c \in A_1$ which means $t_{k-1}(A_1) \geq t_{k-1}(\{c\})$. Since $c \in \tau(A_k, t_{k-1})$, we have $t_{k-1}(\{c\}) = t_{k-1}(A_k)$. This implies that $t_{k-1}(A_1) \geq t_{k-1}(A_k)$. Using (4) this means $t_{k-1}(A_1) > t_k(A_k)$. Since ϵ is arbitrarily small, we have $t_{k-1}(A_1) > t_k(A_k) + \epsilon = t_{k+1}(c)$. Moreover, as $t_{k-1}(A_1) = t_k(A_1) = \alpha$, we have $t_k(A_1) > t_{k+1}(c)$. Now using Claim 3.1 we get $t_{k+1}(A_1) \geq t_k(A_1) >$

$t_{k+1}(\{c\})$. This proves that $c \notin \tau(A_1, t_{k+1})$. This completes the proof of the Claim 3.2.

We now complete the proof by showing that f violates $(k-1)$-cycle or lower cycle monotonicity for every possible outcome at t_{k+1}. Note that for any alternative $X \in \mathcal{A}$,

$$\begin{aligned} t_{k+1}(X) &= t_k(X) \text{ if } b, c \notin \tau(X, t_{k+1}), \\ t_{k+1}(X) &= t_1(A_2) - \epsilon \text{ or } t_k(A_k) + \epsilon \text{ otherwise.} \end{aligned}$$

In the following we consider all the above possibilities case by case.

Case 1. Consider an alternative X such that $t_{k+1}(X) = t_k(X)$. Suppose $f(t_{k+1}, t_{-i}) = X$. Note that since $c \in A_k$, $t_{k+1}(A_k) \geq t_{k+1}(\{c\}) = t_k(A_k) + \epsilon > t_k(A_k)$. Hence,

$$t_{k+1}(X) + t_k(A_k) < t_k(X) + t_{k+1}(A_k),$$

which means f violates 2-cycle monotonicity over t_k and t_{k+1}. Hence $f(t_{k+1}, t_{-i}) \neq X$.

Case 2. Consider an alternative X such that $t_{k+1}(X) \neq t_k(X)$. Then $t_{k+1}(X)$ is either $t_1(A_2) - \epsilon$ or $t_k(A_k) + \epsilon$. We distinguish cases where $t_1(A_2) - \epsilon \neq t_k(A_k) + \epsilon$ and $t_1(A_2) - \epsilon = t_k(A_k) + \epsilon$.

Case 2.1. Suppose $t_1(A_2) - \epsilon \neq t_k(A_k) + \epsilon$.

Case 2.1.1. Suppose $t_{k+1}(X) = t_1(A_2) - \epsilon$. This means $b \in X$. As $b \in \tau(A_2, t_1)$, $t_1(X) \geq t_1(A_2)$. Suppose $f(t_{k+1}, t_{-i}) = X$. Then,

$$t_{k+1}(X) + t_1(A_1) < t_1(A_2) + t_{k+1}(A_1) \leq t_1(X) + t_{k+1}(A_1),$$

which means f violates 2-cycle monotonicity over t_1 and t_{k+1}. Hence $f(t_{k+1}) \neq X$.

Case 2.1.2. Suppose $t_{k+1}(X) = t_k(A_k) + \epsilon$. This means $c \in X$. Using $(k-1)$-cycle monotonicity of f over $t_2, t_3, \ldots, t_{k-1}, t_{k+1}$ we have

$$t_2(A_2) + t_3(A_3) + \ldots + t_{k-1}(A_{k-1}) + t_{k+1}(X) \geq t_{k+1}(A_2) + t_2(A_3) + \ldots + t_{k-1}(X).$$

As $b \in \tau(A_2, t_1)$ we have $t_{k+1}(A_2) \geq t_{k+1}(\{b\}) = t_1(\{b\}) - \epsilon = t_1(A_2) - \epsilon$. Moreover as $c \in X$ and $c \in \tau(A_k, t_{k-1})$, $t_{k-1}(X) \geq t_{k-1}(c) = t_{k-1}(A_k)$. Using all these in the above equation we have

$$t_2(A_2) + t_3(A_3) + \ldots + t_{k-1}(A_{k-1}) + t_k(A_k) + \epsilon \geq t_1(A_2) - \epsilon + t_2(A_3) + \ldots + t_{k-1}(A_k).$$

Since ϵ is arbitrarily small, this violates (3). Hence, $f(t_{k+1}, t_{-i}) \neq X$.

Case 2.2. Suppose that $t_{k+1}(X) = t_1(A_2) - \epsilon = t_k(A_k) + \epsilon$. This means either b or c is in X. Hence $f(t_{k+1}, t_{-i}) \neq X$ by Case 2.1.

It follows from the above consideration that f violates $(k-1)$-cycle or lower cycle monotonicity for every possible outcome at t_{k+1} which is a contradiction to our initial assumption. This completes the proof of Lemma 3.1.

■

Example 3.1. In the following example we show the construction of t_{k+1} for $k = 3$. The circled numbers denote the outcomes at respective types, and $\epsilon = 0.3$ in t_4.

	a	b	c	$max(ab)$	$max(bc)$	$max(ca)$	$max(abc)$
t_1	1	3	(2)	3	3	1	3
t_2	5	3	2	5	(3)	5	5
t_3	4	1	2	4	2	(4)	4
t_4	4.3	2.7	2	4.3	2.7	4.3	4.3

Table 1: Construction of t_{k+1} for $k = 3$

4 Application

In this section we apply our result to a few other interesting domains. Note that in a max domain a set of objects is evaluated as the most valuable object in that set. A more natural assumption would be that the valuation of a set of objects is in-between the maximum valuation and the total valuation of all objects. We call such a domain an intermediate domain. Below we provide the formal definition.

Definition 4.1 (Intermediate Domain)**.** Intermediate domain $\mathcal{I}$ is defined as the domain of types t satisfying the following two conditions: for all $X \in \mathcal{A}$

1. $t(X) \geq t(Y)\ \forall Y \subseteq X$ and
2. $t(X) \leq \sum_{a \in X} t(\{a\})$.

Definition 4.2. A domain $\mathcal{D}$ is convex if $t_1, t_2 \in \mathcal{D}$ implies $\alpha t_1 + (1-\alpha)t_2 \in \mathcal{D}$ for all $\alpha \in [0,1]$.

Note that the lower boundary of an intermediate domain gives a max domain, and the upper boundary is the so called sum domain where the valuation of a set of objects is the total valuation of the objects in that set. It is known that weak monotonicity is sufficient for implementability in the sum domain. Moreover, using our result we know that weak monotonicity is also sufficient for the lower bound of intermediate domain, i.e. the max domain. So, we finally conclude that weak monotonicity is sufficient for implementation for the intermediate domain. This also follows from the fact that intermediate domain is convex, and weak monotonicity is sufficient for implementability in convex domain (Saks and Yu (2005)). Moreover, the additional fact that weak monotonicity is sufficient implementability for both the lower and upper bounds of intermediate domain has serious contribution in finding the optimal mechanism in the intermediate domain. This is because under some mild assumption, the optimal mechanisms always lie in the boundary. We formally state all these facts in the following lemmas and theorems. The proof of the following lemma is left to the reader.

Lemma 4.1. *An intermediate domain is a convex domain.*

We now present a theorem that states that weak monotonicity is sufficient implementability on a convex domain. The proof of this theorem can be found in Saks and Yu (2005).

Theorem 4.1 (Saks and Yu (2005))**.** *Weak monotonicity is sufficient for implementability in convex domain.*

The following corollary follows from both Theorem 3.2 and Theorem 4.1.

Corollary 1. *Weak monotonicity is sufficient for implementability in intermediate domain.*

5 Conclusion

In this paper we show that weak monotonicity is sufficient for implementability in max domain. The proof strategy is completely new in this literature and flexible enough to be extended to prove similar results in other domains of similar structure. One such important domain is the gross substitute domain. We are working on that problem.

References

[1] Itai Ashlagi, Mark Braverman, Avinatan Hassidim, and Dov Monderer. Monotonicity and Implementability. *Econometrica*, 78:1749–1772, 2010.

[2] S. Bikhchandani, S. Chatterji, R. Lavi, A. Mualem, N. Nisan, and A. Sen. Weak monotonicity characterizes deterministic dominant strategy implementation. *Econometrica*, 74:1109–1132, 2006.

[3] Debasis Mishra and Souvik Roy. Implementation in multidimensional dichotomous domains. *Theoretical Economics*, 8:431–466, 2013.

[4] Debasis Mishra, Anup Pramanik, and Souvik Roy. Multidimensional mechanism design in single peaked type spaces. *Journal of Economic Theory*, 153:103–116, 2014.

[5] Roger B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6:58–73, 1981.

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

[7] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970.

[8] M. E. Saks and L. Yu. Weak Monotonicity Suffices for Truthfulness on Convex Domains. In *Proceedings of* 7^{th} *ACM Conference on Electronic Commerce*, pages 286–293. ACM Press, 2005.