Abstract

We consider implementation of a deterministic allocation rule using transfers in quasi-linear private values environments. We show that if a multidimensional type space satisfies some ordinal restrictions, then an allocation rule is implementable in such a type space if and only if it satisfies a familiar and simple condition called 2-cycle monotonicity. Our ordinal restrictions cover type spaces which are non-convex, e.g., the single peaked type space and its generalizations. If we exclude indifferences in our types or use a mild continuity condition, then our result holds in a larger class of type spaces. Thus, we uncover an important role of indifferences in such characterizations.

Keywords. implementation, 2-cycle monotonicity, revenue equivalence, local incentive compatibility, single-peakedness.

JEL codes. D44, D47, D71, D82, D86.
1 Introduction

An enduring theme in mechanism design is to investigate conditions that are necessary and sufficient for implementing an allocation rule. We investigate this question in private values and quasi-linear utility environments when the set of alternatives is finite and the allocation rule is deterministic (i.e., does not randomize). 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 large class of multidimensional type spaces that satisfy some ordinal restrictions, implementability is equivalent to a simple condition called 2-cycle monotonicity. By virtue of revenue equivalence, which holds in these type spaces, we are able to characterize the entire class of dominant strategy incentive compatible mechanisms. The 2-cycle 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

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

where for any alternative $x$, $t(x)$ and $s(x)$ denote the values of alternative $x$ in types $t$ and $s$ respectively.

One of the earliest papers to pursue this question was Rochet (1987), who proved a very general result. He showed that a significantly stronger condition called cycle monotonicity is necessary and sufficient for implementability in any type space - see also Rockafellar (1970). Myerson (1981) formally establishes that in the single object auction set up, where the type is single dimensional, 2-cycle monotonicity is necessary and sufficient for implementation (in Myerson’s set up, this is true even if we consider randomized allocation rules) - see also Spence (1974); Mirrlees (1976). When the type space is multidimensional, if the set of alternatives is finite and the type space is convex, 2-cycle monotonicity implies cycle monotonicity (Bikhchandani et al., 2006; Saks and Yu, 2005; Ashlagi et al., 2010). Though convexity is a natural geometric property satisfied in many economic environments, it excludes many interesting type spaces. Moreover, how far this result extends to type spaces that do not satisfy convexity remain an intriguing question - we discuss this issue in detail in Section 2. A primary objective of this paper is to formulate restrictions on type spaces without the convexity assumption made in the literature and answer the question of implementability in such multidimensional type spaces. Indeed, our restrictions allow many interesting multidimensional non-convex type spaces. Prominent type spaces covered by our formulation are the single peaked type spaces \(^1\) and its generalizations. In all these type spaces, we show that

\(^1\)Roughly, a single peaked type is defined using a strict and complete order on the set of alternatives. A
2-cycle monotonicity is necessary and sufficient for implementability. To our knowledge, this paper is the first to identify such a large class of interesting non-convex type spaces where 2-cycle monotonicity characterizes implementability.

We use a novel method to impose ordinal restriction on type spaces. Such a method of imposing ordinal restriction is usually followed in the mechanism design literature without transfers (a la strategic voting or social choice theory literature). To see how such restrictions can be imposed in a cardinal environment like ours, note that a type in our environment is a vector in $\mathbb{R}^{|A|}$, where $A$ is the set of alternatives. Now, let us restrict attention to strict types, where value of no two alternatives is the same. Such a type must induce a complete and strict ordering on $A$. We put restrictions by allowing only a subset of orderings that can be induced by any type. We discuss such restrictions in detail in Section 3. The set of all strict types in $\mathbb{R}^{|A|}$ that induce an ordering belonging to a set of permissible orderings define a strict type space. To allow for indifferences, we take the closures of such type spaces. We call such type spaces ordinal type spaces. A prominent ordinal type space where our 2-cycle monotonicity characterization holds is the type space induced by all single peaked preference ordering on a tree graph, where the graph consists of alternatives as nodes and a preference ordering must be single peaked along paths of the tree. Single peakedness on a tree is a generalization of classical single peaked preference orderings due to Demange (1982). A detailed definition is given in Section 3. We also show that for a large class of permissible preference orderings, the 2-cycle monotonicity characterization holds if the type space consists of only strict types induced by these orderings. We give an example to show that our result does not hold if we allow for indifference in such type spaces. If we assume a mild continuity condition on the allocation rules, then our result extends to such type spaces when we allow for indifferences. Thus, we highlight an important consequence of restricting attention to only strict type spaces.

Though we identify many different ordinal type spaces, the proof methodology we employ for them is quite similar. This shows that our general methodology is quite robust and can be potentially applied to other type spaces that we do not discuss in the paper.

A characterization of implementability using 2-cycle monotonicity is useful because the cycle monotonicity condition, which can be used to characterize implementability in any type space, is a difficult condition to use and interpret. On the other hand, 2-cycle monotonicity is a simpler condition and the appropriate extension of the monotonicity condition used by Myerson (1981) to characterize implementability in the single object auction model. For this reason, 2-cycle monotonicity is often referred to as weak monotonicity (Bikhchandani et al., 2006; Saks and Yu, 2005) or monotonicity (Ashlagi et al., 2010). In his paper, Rochet (1987) type is single peaked if the values of alternatives decrease as we go to the left or right (where left and right are defined with respect to the given order) of the peak (the highest valued alternative).
likens the implementability question to the rationalizability question in revealed preference theory. Quoting Rochet:

Condition (3) \(^2\) is thus the analogue of the Strong Axiom of Revealed Preferences (SARP), and our theorem is the analogue of Afriat’s result [Afriat, (1965)], which shows how to compute, for any set of data satisfying SARP, a utility function which rationalizes the data. In the one dimensional context, one can restrict oneself to cycles of order 2: condition (3) for 2-cycles is the analogue of the Weak Axiom of Revealed Preferences (WARP).

We also characterize the set of payment rules that can implement an implementable allocation rule. We do this by establishing revenue equivalence in a large class of ordinal type spaces. Revenue equivalence is a property which stipulates that two payment rules that implement the same allocation rule must differ by a constant. We show that the revenue equivalence result holds in a much larger class of ordinal type spaces than the 2-cycle monotonicity result. \(^3\) By characterizing the implementable allocation rules using 2-cycle monotonicity and payments by revenue equivalence, we characterize the set of dominant strategy incentive compatible mechanisms in our multidimensional type spaces with ordinal restrictions.

An important objective in mechanism design is to design expected revenue maximizing mechanisms. While Myerson (1981) solved this problem for the sale of a single object, the problem remains unsolved for multidimensional problems - see a recent take on this topic in Manelli and Vincent (2007); Hart and Nisan (2012); Hart and Reny (2012). However, as Myerson illustrates, there are two important steps in solving the optimal auction problem: (a) characterizing the implementable allocation rules using a monotonicity property and (b) establishing revenue equivalence to pin down the payments. Though the eventual optimization problem remains illusive in the multidimensional type spaces, the literature has made significant progress in advancing these two steps for multidimensional type spaces. Our results add to this literature and we hope that these advances will eventually help us solve the revenue maximization problem in the multidimensional type spaces.

The rest of the paper is organized as follows. In Section 2, we define the model. In Section 3, we define the single peaked type space on a tree and state our main result. Section 4 introduces the idea of ordinal type spaces consisting of strict types where the 2-cycle monotonicity characterization holds. We describe the set of payment rules that implement an implementable allocation rule in Section 6 using revenue equivalence. Section 5 introduces another ordinal type space where the methodology used in our earlier proofs

\(^2\)Condition (3) in Rochet (1987) is the cycle monotonicity condition.

\(^3\)The exact connection of our revenue equivalence result with the literature is established later.
can be used to derive the 2-cycle monotonicity characterization. We relate our results to the literature and conclude in Section 7. The Appendix contains all omitted proofs.

2 Implementation and Cycle Monotonicity

We consider a model with a single agent. As is well known in this literature, this is without loss of generality. All our results generalize easily to a model with multiple agents. The single agent is denoted by $i$. The set of alternatives for agent $i$ is denoted by $A$. In an $n$-agent model, $A$ denotes the possible allocations of agent $i$. The type (private information) of agent $i$ is a vector $t \in \mathbb{R}^{|A|}$. If agent $i$ has type $t$, then $t(a)$ will denote the value of agent $i$ for alternative $a$. We assume private values and quasi-linear utility. This means that if alternative $a$ is chosen and agent $i$ with type $t$ makes a payment of $p$, then his net utility is given by $t(a) - p$.

Not all possible vectors in $\mathbb{R}^{|A|}$ can be a type of agent $i$. Let $D \subseteq \mathbb{R}^{|A|}$ be the type space of agent $i$ - these are the permissible types of agent $i$. An allocation rule is a mapping $f : D \rightarrow A$. We will assume that $f$ is onto. This is standard in the literature - if $f$ is not onto, then all the results can be restated in terms of range of $A$.

A payment rule of agent $i$ is a mapping $p : D \rightarrow \mathbb{R}$. A mechanism consists of an allocation rule and a payment rule.

**Definition 1** An allocation rule $f$ is implementable if there exists a payment rule $p$ such that for every $s, t \in D$, we have

$$s(f(s)) - p(s) \geq s(f(t)) - p(t).$$

In this case, we will say that $p$ implements $f$ and $(f, p)$ is an incentive compatible mechanism.

The primary objective of this paper is to give a simple necessary condition on the allocation rule that is also sufficient for implementability in a large class of interesting type spaces. For this, we revisit a classic condition that is already known to be necessary and sufficient for implementability in any type space.

**Definition 2** An allocation rule $f$ is $K$-cycle monotone, where $K \geq 2$ is a positive integer, if for every finite sequence of types $(t^1, t^2, \ldots, t^k)$, with $k \leq K$, we have

$$\sum_{j=1}^{k} [t^j(f(t^j)) - t^j(f(t^{j-1}))] \geq 0,$$

(1)

For instance, in a model with $n$ agents and $n$ objects, where each agent can be assigned exactly one object and there is no externality in allocations across agents, $A$ will be the set of objects and not the set of matchings.
where \( t^0 \equiv t^k \). An allocation rule \( f \) is **cyclically monotone** if it is \( K \)-cycle monotone for every positive integer \( K \geq 2 \).

It is well known that implementability is equivalent to cycle monotonicity (Rochet, 1987; Rockafellar, 1970). This result is very general - it works on any type space \( D \) and does not even require \( A \) to be finite. However, cycle monotonicity is a difficult condition to use and interpret since it requires verifying non-negativity of Inequality 1 for arbitrary length sequences of types. In a series of papers, it has been established that a significantly weaker condition than cycle monotonicity is sufficient for implementation in various interesting type spaces. Bikhchandani et al. (2006) showed that 2-cycle monotonicity is sufficient for implementability if \( D \) is an order-based type space - this includes many interesting type spaces in the context of multi-object auctions. Saks and Yu (2005) show that 2-cycle monotonicity is sufficient for implementation if \( D \) is convex - this extends the result in Bikhchandani et al. (2006) because an order-based type space is convex. Ashlagi et al. (2010) extend this result to show that if the closure of \( D \) is convex, then 2-cycle monotonicity is sufficient for implementation.

However, Mishra and Roy (2013) show that there are interesting non-convex type spaces where 2-cycle monotonicity is not sufficient for implementation. Further, they identify an interesting class of non-convex type spaces where 3-cycle monotonicity is sufficient for implementation but 2-cycle monotonicity is not sufficient.

Interestingly, Ashlagi et al. (2010) establish a surprising result by allowing for randomization, i.e., an allocation rule picks a probability distribution over alternatives. They show that if every 2-cycle monotone randomized allocation rule is also cyclically monotone in a type space \( D \) of dimension at least 2, then the closure of \( D \) must be convex.

It is not clear how far this result is true if \( f \) is allowed to be deterministic. Vohra (2011) contains a simple example of a non-convex type space with four alternatives where every deterministic allocation rule satisfying 2-cycle monotonicity is implementable. In his example, Vohra (2011) considers the sale of two objects \( \alpha \) and \( \beta \) to agents. The set of alternatives is the set of all subsets of \( \{\alpha, \beta\} \). The restriction on values of agents is the following: \( t(\{\alpha, \beta\}) = \max(t(\{\alpha\}), t(\{\beta\})) \) and \( t(\emptyset) = 0 \). Hence, each agent desires at most one object, though he may be assigned both the objects. The type space here is non-convex. To see this, consider two types of agent \( i \)

\[
\begin{align*}
t(\emptyset) &= 0, \quad t(\{\alpha\}) = 3, \quad t(\{\beta\}) = 4, \quad t(\{\alpha, \beta\}) = 4 \\
s(\emptyset) &= 0, \quad s(\{\alpha\}) = 5, \quad s(\{\beta\}) = 4, \quad s(\{\alpha, \beta\}) = 5.
\end{align*}
\]

\( ^5 \) When the set of alternatives is finite, this result can be slightly strengthened to say that implementability is equivalent to \(|A|\)-cycle monotonicity (Mishra and Roy, 2013).
A convex combination of (0.5, 0.5) of these two types generates values 4 for objects $\alpha$ and $\beta$ but a value of 4.5 for the bundle of objects $\{\alpha, \beta\}$. This violates the restriction on the type space.

Note that if we allow at most one object to be assigned to an agent, then the type space becomes convex, and we can apply earlier result to conclude that 2-cycle monotonicity is sufficient for implementation. However, by allowing the alternative $\{\alpha, \beta\}$, but still having a restriction that agents desire at most one object, we get to a non-convex type space. The result in Vohra (2011) shows that 2-cycle monotonicity is sufficient for implementation in such an example. It is not clear on how to extend the proof of this example if there are more than two objects.

2.1 A Motivating Example

Since the type space in the example in Vohra (2011) seems to be a slight modification of a convex type space, it is still unclear whether there are interesting non-convex type space where 2-cycle monotonicity is sufficient for implementation. The result in Ashlagi et al. (2010) shows that if every 2-cycle monotone randomized allocation rule is implementable in a multidimensional type space, then it must be convex. This shows that there is a significant gap in understanding implementability of deterministic allocation rules in non-convex multidimensional type spaces. We give below a motivating example to show that there are interesting non-convex type spaces where the current results are silent. Our results in the paper will apply to such type spaces.

Consider a general scheduling problem as follows. A number of firms procure products/parts from a supplier over a time horizon. In each time period, the supplier can only supply to one firm. Every firm $i$ has a time period $\tau^*$ where it gets the maximum value from getting its products supplied. The firms have single peaked preference over time, i.e., for any time periods $\tau, \tau'$, if $\tau < \tau' < \tau^*$ or $\tau > \tau' > \tau^*$, then a firm values supply of its products at time period $\tau'$ to time period $\tau$ (this may be due to inventory carrying cost and delivery delay costs).

The type space in this example is non-convex. To see this, suppose there are just three time periods $\{1, 2, 3\}$ and consider two single peaked types of an agent (firm): $s := (6, 4, 3)$ (peak value is period 1) and $t := (3, 4, 6)$ (peak value is period 3). A convex combination \[
\frac{s + t}{2}
\] produces the type $(4.5, 4, 4.5)$, which is no longer single peaked.

There are other problems where one encounters single peaked preferences. For instance, consider an agent who is being sold multiple products/objects, but the agent is interested to buy at most one object. Each object has a quality and a maintenance cost, which depends on the quality. The value for an object depends on the trade off between maintenance
cost due to quality and value for quality. Under reasonable assumptions on these cost and
value functions, one gets single peaked preferences, i.e., an optimum level of quality such
that below and above that quality level, value for the objects decline. If we consider the
standard multi-object auctions in this framework, one gets a restriction on type space that
types have to be single peaked.

In such non-convex type spaces, we characterize implementability using 2-cycle mono-
tonicity and apply revenue equivalence to obtain a complete characterization of dominant
strategy incentive compatible mechanisms. Thus, there are interesting type spaces where
earlier results are silent and our results provide sharp characterizations of implementability
and incentive compatibility.

3 The Single Peaked Type Space on a Tree

We start off by considering the problem of choosing an alternative (a location) over a tree
network. Our network $G$ is given by a finite set of nodes $A$ and a set of undirected edges
$E$ between these nodes. The set $A$ is the set of alternatives or outcomes from which one of
the alternatives must be chosen. We will assume that $G$ is a tree, i.e., a graph whose edges
do not form any cycles and there is a unique path between every pair of alternatives/nodes.
The private information or type of each agent is a vector $t \in \mathbb{R}^{|A|}$, where $t(a)$ denotes the
value for alternative $a$ at this type. The set of possible types (type space) of each agen
twill be determined by $G$.

We define the type space by imposing ordinal restriction on type spaces. Notice that each
type induces a weak ordering on the set of alternatives. We call a type $t$ strict if $t(a) \neq t(b)$
for all $a \neq b$. A strict type induces a linear order on the set of alternatives. Let $\mathcal{P}$ be the
set of all linear orders over $A$. Given a linear order $P \in \mathcal{P}$, we denote the $k$-th ranked
alternative in $P$ as $P(k)$. Given any pair of alternatives $a, b \in A$, there exists a unique
path in $G$ between $a$ and $b$, and we denote this unique path as $\Pi(a, b)$. With a slight abuse
of notation, we let $\Pi(a, b)$ to denote also the set of alternatives (including $a$ and $b$) in the
unique path from $a$ to $b$ in $G$. A linear order $P \in \mathcal{P}$ is single peaked with respect to $G$
if for every $a \in A$ and every $b \neq a$ but $b \in \Pi(a, P(1))$, we have $bPa$. Let $\mathcal{D} \subseteq \mathcal{P}$ be the
set of all single peaked linear orders in $\mathcal{P}$. If $G$ is a line graph, this reduces to the standard
definition of single peaked preference ordering. This generalization of single peakedness is
due to Demange (1982).

**Definition 3** The strict single peaked type space $T^G$ (with respect to $G$) is the set of

---

6For instance, if an object with quality $q$ generates a net value of $vq - q^2$, where $v$ is per unit value of
quality and $q^2$ is the maintenance cost, then the optimal quality level is $\frac{v}{2}$. 

---

8
all non-negative type vectors that induce a linear order in \( \mathcal{D} \), i.e.,

\[ T^G := \{ t \in \mathbb{R}^{|A|}_+ : t \text{ induces } P \text{ for some } P \in \mathcal{D} \}. \]

The **single peaked type space** is \( cl(T^G) \), where \( cl(T^G) \) denotes the closure of the set \( T^G \).

The main result of this section is the following.

**Theorem 1** An allocation rule \( f : cl(T^G) \rightarrow A \) is implementable if and only if it is 2-cycle monotone.

**Remark.** In many contexts, it is natural to assume that there is an alternative whose value is always zero (for instance, in auction problems, the alternative of not getting any object gives zero value to the agent). Though we do not explicitly allow this in our model, all our proofs can be modified straightforwardly to accommodate the fact that there is an alternative which is worst ranked and has value zero at every type.

### 3.1 Proof of Theorem 1

The proof of Theorem 1 will be done using a series of Lemmas. These lemmas will reveal the underlying structure of the type space. Further, we will show how these steps can be used in other type spaces to extend Theorem 1.

Denote by \( D \equiv cl(T^G) \). First, by Rochet (1987), if \( f : D \rightarrow A \) is implementable, then it is 2-cycle monotone. Next, again by Rochet (1987), if \( f \) is cyclically monotone, then it is implementable. So, we will show that if \( f \) is 2-cycle monotone, then it is cyclically monotone. In the remainder of the section, we assume that \( f \) is 2-cycle monotone.

For every \( a \in A \), define \( D(a) \) as follows.

\[ D(a) := \{ t \in D : f(t) = a \}. \]

Since \( f \) is onto, \( D(a) \) is non-empty. Next, for every \( s, t \in D \), define \( \ell(s, t) \) as follows.

\[ \ell(s, t) := t(f(t)) - t(f(s)). \]

Notice that 2-cycle monotonicity is equivalent to requiring that for every \( s, t \in D \), we have \( \ell(s, t) + \ell(t, s) \geq 0 \). Now, for every \( a, b \in A \), define \( d(a, b) \) as follows.

\[ d(a, b) := \inf_{t \in D(b)} [t(b) - t(a)]. \]

We state below a well known fact - see, for instance, Lemma 6 in Bikhchandani et al. (2006).

**Lemma 1** For every \( a, b \in A \), \( d(a, b) + d(b, a) \geq 0 \).
Proof: Suppose \(d(a, b) + d(b, a) = -\epsilon < 0\) for some \(a, b \in A\). This means, there is a \(s \in D(b)\) and \(t \in D(a)\) such that \(s(b) - s(a) + [t(a) - t(b)] < 0\). But this means that \(\ell(s, t) + \ell(t, s) < 0\), a contradiction to 2-cycle monotonicity. \(\blacksquare\)

For any \(a, b \in A\), we say \(a\) and \(b\) are \textbf{G-neighbors} if the unique path between \(a\) and \(b\) in \(G\) is a direct edge between \(a\) and \(b\) in \(G\). The following facts will be useful throughout the proofs. These facts are true due to the single peakedness of the type space.

**Fact 1** For any \(a, b \in A\), if \(a\) and \(b\) are \(G\)-neighbors, then there exists a linear order \(P \in D\) such that \(P(1) = a, P(2) = b\).

Fact 1 says that if \(a\) and \(b\) are \(G\)-neighbors then there is some ordering where they are ranked first and second.

**Fact 2** For any \(a, c \in A\) and \(b \in \Pi(a, c)\) such that \(b\) is a \(G\)-neighbor of \(a\), there exists a linear order \(P \in D\) such that \(\{P(1), P(2)\} = \{a, b\}\), \(xPc\) for all \(x \in \Pi(a, c) \setminus \{c\}\), and \(cPx\) for all \(x \notin \Pi(a, c)\).

Fact 2 says that if \(a\) and \(c\) are any pair of alternatives with \(b\) being a \(G\)-neighbor of \(a\) in \(\Pi(a, c)\), then there is some ordering where \(a\) and \(b\) are first and second ranked, followed by all the other alternatives in \(\Pi(a, c)\), and followed by the remaining alternatives outside \(\Pi(a, c)\). The first step of the proof of Theorem 1 is the following lemma.

**Lemma 2** If \(a, b\) are \(G\)-neighbors, then \(d(a, b) + d(b, a) = 0\).

Proof: Consider \(a, b \in A\) such that \(a\) and \(b\) are \(G\)-neighbors. By Lemma 1, \(d(a, b) + d(b, a) \geq 0\). Assume for contradiction \(d(a, b) + d(b, a) = \epsilon > 0\). Then, either \(d(a, b) > \frac{\epsilon}{2}\) or \(d(b, a) > \frac{\epsilon}{2}\). Suppose \(d(a, b) > \frac{\epsilon}{2}\) - a similar proof works if \(d(b, a) > \frac{\epsilon}{2}\). Then, there is a type \(s \in D(b)\) such that \(d(a, b) \leq s(b) - s(a) < d(a, b) + \epsilon_1\), for any \(\epsilon_1 > 0\) arbitrarily close to zero, in particular \(\epsilon_1 < \frac{\epsilon}{2}\). Hence, \(s(b) - s(a) > \frac{\epsilon}{2}\). We now choose a \(\delta \in (2\epsilon_1, s(b) - s(a))\) but arbitrarily close to \(2\epsilon_1\). Since \(a\) and \(b\) are \(G\)-neighbors, by Fact 1, there exists a \(P \in D\) such that \(b\) is top ranked and \(a\) is second ranked. We can construct a type \(u \in D\) that induces \(P\) and

\[
u(x) = \begin{cases} 
s(x) + \delta & \text{if } x = a \\
s(x) + \frac{\delta}{2} & \text{if } x = b \\
\leq \min(s(x), s(a)) & \text{if } x \notin \{a, b\},
\end{cases}
\]

Notice that since \(s(b) > s(a)\), we have \(u(b) > u(a)\) for sufficiently small \(\delta > 2\epsilon_1\). Also, alternatives other than \(a\) and \(b\) are ordered according to \(P\) but their values are not increased.

We will now argue that \(f(u) = a\). First, if \(f(u) = x \notin \{a, b\}\), we have \(u(x) - u(b) \leq s(x) - s(b) - \frac{\delta}{2} < s(x) - s(b)\), which violates 2-cycle monotonicity. Second, if \(f(u) = b\), we
have \( u(b) - u(a) = s(b) - s(a) - \frac{\epsilon}{2} < d(a, b) - \left(\frac{\epsilon}{2} - \epsilon_1\right) < d(a, b) \), which violates the definition of \( d(a, b) \). Hence, \( f(u) = a \).

But this implies that \( d(b, a) \leq u(a) - u(b) = s(a) - s(b) + \frac{\epsilon}{2} \leq -d(a, b) + \frac{\epsilon}{2} \). Hence, \( d(b, a) + d(a, b) \leq \frac{\epsilon}{2} \). Since \( \delta, \epsilon_1 \) can be chosen arbitrarily close to zero, this contradicts the fact that \( d(a, b) + d(b, a) = \epsilon > 0 \).

The next step is to show that for any pair of alternatives \( a \) and \( c \), there is some alternative \( b \in \Pi(a, c) \) such that a version of the reverse triangle inequality holds between \( a, b \), and \( c \) using \( d(\cdot, \cdot) \).

**Lemma 3** For any pair of alternatives \( a, c \in A \) such that \( a \) and \( c \) are not \( G \)-neighbors, there exists an alternative \( b \in \Pi(a, c) \) such that

\[
d(a, b) + d(b, c) \leq d(a, c).
\]

**Proof:** Fix \( a, c \in A \) such that \( a \) and \( c \) are not \( G \)-neighbors. Choose an \( \epsilon > 0 \) and arbitrarily close to zero and a \( t \in D(c) \) such that \( d(a, c) \leq t(c) - t(a) \leq d(a, c) + \epsilon \). We consider two cases.

**Case 1.** \( t(c) \geq t(a) \). Choose \( b \in \Pi(a, c) \) such that \( b \) is a \( G \)-neighbor of \( c \). By single peakedness, for every \( x \in \Pi(a, c) \), we have \( t(x) \geq t(a) \). Then, we can construct a new type in which \( b \) and \( c \) occupy the top two ranks. We construct such a new type \( s \) as follows. Choose \( \epsilon' > 0 \) but arbitrarily close to zero and let \( \delta := t(c) - t(b) - d(b, c) + 2\epsilon' \). Note that since \( t \in D(c) \), we have \( t(c) - t(b) \geq d(b, c) \), and this implies that \( \delta > 0 \).

\[
s(x) = \begin{cases} 
  t(x) + \epsilon' & \text{if } x = c \\
  t(a) & \text{if } x \in \Pi(a, c) \setminus \{b, c\} \\
  t(x) + \delta & \text{if } x = b \\
  \leq \min(t(x), t(a)) & \text{if } x \notin \Pi(a, c).
\end{cases}
\]

By Fact 2, we can define \( s \) such that it is in \( cl(T^{G}) \). We argue that \( f(s) = b \). First, suppose \( f(s) = x \notin \{b, c\} \). Then, \( s(x) - s(c) < t(x) - t(c) \), and this contradicts 2-cycle monotonicity. Next, suppose \( f(s) = c \). Then, \( d(b, c) \leq s(c) - s(b) = t(c) - t(b) - \delta + \epsilon' = d(b, c) - \epsilon' < d(b, c) \), a contradiction. Hence, \( f(s) = b \).

Now, \( d(a, b) \leq s(b) - s(a) = [t(b) - t(a) + \delta] = t(c) - t(a) - d(b, c) + 2\epsilon' \leq d(a, c) - d(b, c) + 2\epsilon' + \epsilon \). Since \( \epsilon \) and \( \epsilon' \) can be chosen arbitrarily close to zero, we conclude that \( d(a, b) + d(b, c) \leq d(a, c) \).

**Case 2.** \( t(c) < t(a) \). Let \( b \in \Pi(a, c) \) be the \( G \)-neighbor of \( a \). Define the subset of alternatives \( C \) as follows: \( C := \{c' \in \Pi(b, c) : t(c') = t(c) \text{ and } \forall c'' \in \Pi(c', c), t(c'') = t(c)\} \).
In other words, $C$ is the set of “contiguous” alternatives in $\Pi(b, c)$ starting from $c$ which have the same value as $t(c)$ in type $t$. Now, construct a new type $s$ as follows. Choose an $\epsilon' > 0$ but arbitrarily close to zero. Note that $\epsilon'$ can be chosen sufficiently close to zero such that for all $x \in \Pi(a, c) \setminus C$, we have $t(x) > t(c) + \epsilon'$. Also, choose $\delta = t(c) - t(b) - d(b, c) + 2\epsilon'$. As before, $\delta > 0$.

$$
s(x) = \begin{cases} 
t(c) + \epsilon' & \text{if } x \in \Pi(a, c) \setminus \{a, b\} \\
t(x) & \text{if } x = a \\
t(x) + \delta & \text{if } x = b \\
\leq \min(t(x), s(c)) & \text{if } x \notin \Pi(a, c).
\end{cases}
$$

Again, by Fact 2, such a type $s$ can be found in $cl(T^G)$. If $f(s) = x \notin C \cup \{b\}$, then $s(x) - s(c) < t(x) - t(c)$, which violates 2-cycle monotonicity. Hence, $f(s) \in C \cup \{b\}$. If $f(s) = c$, then $d(b, c) \leq s(c) - s(b) = t(c) - t(b) - \delta + \epsilon' = d(b, c) - \epsilon' < d(b, c)$, a contradiction. So, $f(s) \in (C \setminus \{c\}) \cup \{b\}$. We consider two subcases.

**Case 2a.** Suppose $f(s) = c' \in C \setminus \{c\}$. Then, $d(c, c') \leq s(c') - s(c) = 0$. But $f(t) = c$ implies that $d(c', c) \leq t(c) - t(c') = 0$. This implies that $d(c, c') + d(c', c) \leq 0$. By Lemma 1, $d(c, c') + d(c', c) = 0$. Since, $d(c, c') \leq 0$ and $d(c', c) \leq 0$, we conclude that $d(c, c') = d(c', c) = 0$. Further, since $f(s) = c'$, $d(a, c') \leq s(c') - s(a) = t(c) - t(a) + \epsilon' \leq d(a, c) + \epsilon + \epsilon'$. Since $\epsilon$ and $\epsilon'$ can be chosen arbitrarily small, $d(a, c') \leq d(a, c)$. Hence, $d(a, c') + d(c', c) \leq d(a, c)$, where we used the fact that $d(c', c) = 0$. This completes the proof of this case.

**Case 2b.** Suppose $f(s) = b$. Then, $d(a, b) \leq s(b) - s(a) = t(b) - t(a) + \delta = t(c) - t(a) - d(b, c) + 2\epsilon' \leq d(a, c) - d(b, c) + 2\epsilon' + \epsilon$. Since $\epsilon$ and $\epsilon'$ can be chosen arbitrarily small, $d(a, b) + d(b, c) \leq d(a, c)$. This completes the proof of this case. 

Lemmas 2 and 3 are the foundations of our proof. The next lemma (and many subsequent lemmas) is a consequence of these two lemmas. Lemmas 2 and 3 are the only place where we use the fact that the type space is $cl(T^G)$. This implies that as long as we can prove analogues of Lemmas 2 and 3 in a type space, Theorem 1 continues to hold. Now, consider the following lemma.

**Lemma 4** For any pair of alternatives $a_1, a_k \in A$, let $\Pi(a_1, a_k) = (a_1, a_2, \ldots, a_k)$ with $k > 2$. Then, the following are true.

$$
d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{k-1}, a_k) \leq d(a_1, a_k)
$$
$$
d(a_k, a_{k-1}) + d(a_{k-1}, a_{k-2}) + \ldots + d(a_2, a_1) \leq d(a_k, a_1).
$$
The set of nodes in Heydenreich et al. (2009) is a pair of alternatives where a Lemma 5: Consider any pair of alternatives \(a, b \in A\) and let \((a_1, a_2, \ldots, a_k)\) be the sequence of alternatives on \(\Pi(a_1, a_k)\). We do the proof using induction on \(k\). If \(k = 3\), then the claim is true due to Lemma 3. Suppose the claim is true for all \(k < K\). If \(k = K\), then by Lemma 3, there is an alternative \(a_r \in \{a_2, \ldots, a_{K-1}\}\) such that \(d(a_1, a_r) + d(a_r, a_K) \leq d(a_1, a_K)\). The paths \((a_1, \ldots, a_r)\) and \((a_r, \ldots, a_K)\) each contain less than \(K\) nodes. By our induction hypothesis, \(d(a_1, a_2) + \ldots + d(a_{r-1}, a_r) \leq d(a_1, a_r)\) and \(d(a_r, a_{r+1}) + \ldots + d(a_{K-1}, a_K) \leq d(a_r, a_K)\). Hence, \(d(a_1, a_2) + \ldots + d(a_{K-1}, a_K) \leq d(a_1, a_K)\).

A similar argument shows that \(d(a_k, a_{k-1} + d(a_{k-1}, a_{k-2}) + \ldots + d(a_2, a_1) \leq d(a_k, a_1)\). \(\blacksquare\)

The following lemma is well known - see, for instance, Heydenreich et al. (2009).

**Lemma 5** Suppose for every sequence of alternatives \((a_1, \ldots, a_k)\), we have

\[
\sum_{j=1}^{k} d(a_j, a_{j+1}) \geq 0,
\]

where \(a_{k+1} \equiv a_1\). Then, \(f\) is cyclically monotone.

**Proof:** Consider any sequence of types \((t^1, \ldots, t^k)\) such that \(f(t^j) = a_j\) for all \(j \in \{1, \ldots, k\}\). Then, \([t^2(a_2) - t^2(a_1)] + \ldots + [t^k(a_k) - t^k(a_{k-1})] + [t^1(a_1) - t^1(a_k)] \geq d(a_1, a_2) + \ldots + d(a_{k-1}, a_k) + d(a_k, a_1) \geq 0\), where we use \(d(a, a) = 0\) for any \(a \in A\). So, \(f\) is cyclically monotone. \(\blacksquare\)

At this point, it will be useful to consider another graph \(G_f\). \(^7\) The set of nodes in \(G_f\) is the set of alternatives \(A\). It is a complete directed graph. Hence, for every pair of alternatives \(a, b \in A\), there is an edge from \(a\) to \(b\) and an edge from \(b\) to \(a\). A path from an alternative \(a\) to another alternative \(b\) in \(G_f\) is a directed path. Note that for every path \((a_1, a_2, \ldots, a_k)\) in \(G_f\) from \(a_1\) to \(a_k\), the corresponding undirected path may or may not exist in \(G\). For any pair of alternatives \(a_1, a_k \in A\), denote by \(\text{dist}_f(a_1, a_k)\) the shortest path length from \(a_1\) to \(a_k\) in \(G_f\).

The next lemmas shows that the shortest path in \(G_f\) between a pair of alternatives \(a\) and \(b\) is the unique path \(\Pi(a, b)\) in \(G\).

**Lemma 6** For any pair of alternatives \(a, b \in A\), let \(\Pi(a, b) \equiv (a \equiv a_1, a_2, \ldots, b \equiv a_k)\). Then,

\[
\sum_{j=1}^{k-1} d(a_j, a_{j+1}) = \text{dist}_f(a, b).
\]

\(^7\)In Heydenreich et al. (2009), this graph is called the *allocation graph.*
Proof: Fix $a, b \in A$ and choose a shortest path from $a$ to $b$ in $G^f$. Note that $G^f$ consists of directed edges and, hence, this path consists of directed edges. Let this path be $(a'_1, \ldots, a'_h)$, where $a'_1 \equiv a$ and $a'_h \equiv b$. Also, there is a unique path $\Pi(a, b)$ between $a$ and $b$ in $G$, and this consists of undirected edges in $G$. We will sometimes refer to this path in $G^f$ by turning them into directed edges - this will give rise to two directed paths, one from $a$ to $b$ and the other from $b$ to $a$. We will denote these two directed paths corresponding to $\Pi(a, b)$ in $G^f$ as $\bar{\Pi}(a, b)$ and $\bar{\Pi}(b, a)$.

Now, take any (directed) edge $(x, y)$ in the path $(a'_1, \ldots, a'_h)$. If $x$ and $y$ are not $G$-neighbors, then we can pick the path $\bar{\Pi}(x, y) \equiv (x, c_1, \ldots, c_r, y)$ from $x$ to $y$, and $d(x, y) \geq d(x, c_1) + d(c_1, c_2) + \ldots + d(c_{r-1}, c_r) + d(c_r, y)$ by Lemma 4. Combining such paths $\bar{\Pi}(a'_j, a'_{j+1})$ for all $j \in \{1, \ldots, h-1\}$, we get the path $\bar{\Pi}(a, b)$ from $a$ to $b$ in $G$, which we denote by $(a_1, \ldots, a_k)$ with $a \equiv a_1$ and $b \equiv a_k$, and some cycles in $G^f$. Since these cycles all consist of edges from $G$, and $G$ does not have a cycle, it must be that these cycles are 2-cycles.

By Lemma 2, these cycles have zero length (according to weights defined in $G^f$). Hence, $\text{dist}^f(a, b) = \sum_{j=1}^{k-1} d(a_j, a_{j+1})$.

This leads to the final lemma in the proof of Theorem 1.

**Lemma 7** Every cycle of $G^f$ has non-negative length.

Proof: Consider a cycle $(a_1, \ldots, a_k, a_1)$ in $G^f$. By Lemma 6, the unique path $\Pi(a_1, a_k) \equiv (a_1, b_1, \ldots, b_r, a_k)$ in $G$ satisfies $d(a_1, b_1) + d(b_1, b_2) + \ldots + d(b_{r-1}, b_r) + d(b_r, a_k) = \text{dist}^f(a_1, a_k) \leq d(a_1, a_2) + \ldots + d(a_{k-1}, a_k)$. This shows that

$$d(a_1, a_2) + \ldots + d(a_{k-1}, a_k) \geq d(a_1, b_1) + d(b_1, b_2) + \ldots + d(b_{r-1}, b_r) + d(b_r, a_k).$$

Now, consider the path $(a_k, b_r, \ldots, b_1, a_1)$ from $a_k$ to $a_1$. By Lemma 4,

$$d(a_k, a_1) \geq d(a_k, b_r) + d(b_r, b_{r-1}) + \ldots + d(b_2, b_1) + d(b_1, a_1).$$

Adding the previous two inequalities, we get

$$\sum_{j=1}^{k} d(a_j, a_{j+1}) \geq [d(a_1, b_1) + d(b_1, a_1)] + [d(b_1, b_2) + d(b_2, b_1)] + \ldots + [d(b_{r-1}, b_r) + d(b_r, b_{r-1})] + [d(a_k, b_r) + d(b_r, a_k)] \geq 0,$$

where $a_{k+1} \equiv a_1$ and the last equality follows from Lemma 2 and the fact that consecutive alternatives on the path $(a_1, b_1, \ldots, b_r, a_k)$ are $G$-neighbors. ■

Lemmas 7 and 5 establish that $f$ is cyclically monotone, and hence, implementable. This completes the proof of Theorem 1.
4 Type Spaces with Ordinal Restrictions: The Difference Indifference Makes

We now investigate how far we can extend Theorem 1. For this, we formally define the notion of an ordinal type space. Let \( \mathcal{D} \subseteq \mathcal{P} \) be some subset of strict orderings of the set of alternatives \( A \). We will refer to \( \mathcal{D} \) as a domain. We denote by \( T(\mathcal{D}) \) the set of all strict types in \( \mathbb{R}_+^{\lvert A \rvert} \) that are consistent with the strict orderings in \( \mathcal{D} \), i.e.,

\[
T(\mathcal{D}) := \{ t \in \mathbb{R}_+^{\lvert A \rvert} : t \text{ is consistent with some } P \in \mathcal{D} \}.
\]

**Definition 4** A type space \( \mathcal{D} \subseteq \mathbb{R}_+^{\lvert A \rvert} \) is ordinal if there exists \( \mathcal{D} \subseteq \mathcal{P} \) such that \( \mathcal{D} = T(\mathcal{D}) \).

In our previous section, we assumed that \( \mathcal{D} \) is the set of all single peaked preferences with respect to a tree graph and showed that if the type space is \( \text{cl}(T(\mathcal{D})) \), then every 2-cycle monotone allocation rule is implementable.

We first give an example to illustrate how tight this result is. Below, we consider an example with three alternatives and \( \mathcal{D} \) that contains one more preference ordering than the set of all single peaked orderings. We show that Theorem 1 fails in this domain.

**Example 1**

Let \( A = \{a, b, c\} \) and \( \mathcal{D} \) consists of the following orderings shown in Table 1.

<table>
<thead>
<tr>
<th></th>
<th>( p_1 )</th>
<th>( p_2 )</th>
<th>( p_3 )</th>
<th>( p_4 )</th>
<th>( p_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>b</td>
<td>c</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>a</td>
<td>c</td>
<td>b</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>c</td>
<td>a</td>
<td>a</td>
<td>b</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: A K-connected domain satisfying top lifting

Notice that \( \mathcal{D} \setminus \{P_5\} \) is the set of all single peaked linear orderings with respect to the graph with edges \( \{a, b\}, \{b, c\}, \{c, a\} \) (i.e., the line graph with \( a \) and \( c \) as endpoints). Let \( \mathcal{D} = T(\mathcal{D}) \). We now define an allocation rule \( f : \text{cl}(\mathcal{D}) \to A \) as follows.

\[
f(t) = \begin{cases}
  a & \text{if } t(a) - t(b) \geq 1, t(a) > t(b) \geq t(c) \\
  b & \text{if } t(a) - t(b) < 1, t(a) \geq t(b) \geq t(c), t(a) \neq t(b) \neq t(c) \\
  b & \text{if } t(b) \geq t(a) \geq t(c), t(a) \neq t(b) \neq t(c) \\
  b & \text{if } t(b) \geq t(c) \geq t(a), t(a) \neq t(b) \neq t(c) \\
  c & \text{if } t(c) \geq t(b) \geq t(a), t(a) \neq t(b) \neq t(c) \\
  c & \text{if } t(c) > t(a) \geq t(b) \\
  a & \text{if } t(c) = t(a) > t(b), t(a) - t(b) \geq 1 \\
  c & \text{if } t(c) = t(a) > t(b), t(a) - t(b) < 1 \\
  c & \text{if } t(a) = t(b) = t(c).
\end{cases}
\]
Note that \(d(a, b) = -1, d(b, a) = 1, d(b, c) = d(c, b) = d(c, a) = d(a, c) = 0\).

Now, to verify if \(f\) is \(K\)-cycle monotone for each \(K\), we use the following equivalent definition of \(K\)-cycle monotonicity. An allocation rule \(f\) is \(K\)-cycle monotone if for every integer \(k \leq K\) and every sequence of alternatives \((a_1, \ldots, a_k)\), we have \(d(a_1, a_2) + \ldots + d(a_{k-1}, a_k) + d(a_k, a_1) \geq 0\). The equivalence of this definition of \(K\)-cycle monotonicity and the earlier definition of \(K\)-cycle monotonicity is well known, and can be found, for instance in (Vohra, 2011; Heydenreich et al., 2009).

Using this, we see that \(f\) is 2-cycle monotone. But \(d(a, b) + d(b, c) + d(c, a) = -1\) implies that \(f\) is not 3-cycle monotone, and hence, not implementable.

### 4.1 Strict Types, Ordinal Connectedness, and Lifting

Surprisingly, if we restrict attention to strict types, Theorem 1 can be shown to be true in a larger class of type spaces (including the interior of the type space discussed in Example 2). In this section, our objective is to define a class of ordinal type spaces consisting of strict types where Theorem 1 continues to hold. \(^8\)

We will focus on ordinal type spaces that are *ordinally connected* in some way. We define two notions of connectedness and a *lifting* property here. As before, we use the notation \(P(k)\) to denote the \(k\)-th ranked alternative in ordering \(P\). Fix a set of preference orderings \(\mathcal{D} \subseteq \mathcal{P}\).

**Definition 5** \(An\) alternative \(a \in A\) can be **top lifted** at an ordering \(P \in \mathcal{D}\), if for every \(b \in A\) with \(aPb\), there exists an ordering \(P' \in \mathcal{D}\) such that (a) \(P'(1) = a\) and (b) if \(bPc\) for any alternative \(c\) then \(bP'c\).

Notice that \(P'\) may be different for different \(b\) in the definition above. Also, if \(P(1) = a\), then \(a\) can be top lifted using \(P\) itself, and hence, the condition is vacuously satisfied.

Table 2 illustrates the idea of top lifting. Suppose \(A = \{x, y, z, x', y', a, b\}\) and \(P\), as shown in Table 2, is in \(\mathcal{D}\). Note that in \(P'\) and \(P''\), \(a\) is the top ranked alternative. For any alternative \(a' \in \{z, b\}\) the alternatives that were worse than \(a'\) in \(P\) continue to be worse than \(a'\) in \(P'\). For any alternative \(a' \in \{x', y'\}\) the alternatives that were worse than \(a'\) in \(P\) continue to be worse than \(a'\) in \(P''\). Hence, if \(P', P'' \in \mathcal{D}\), \(a\) can be top lifted at \(P\).

**Definition 6** A domain of preference orderings \(\mathcal{D}\) satisfies **top lifting** if for every \(a \in A\) and every \(P \in \mathcal{D}\), \(a\) can be top lifted at \(P\).

\(^8\)If the set of preference orderings in \(\mathcal{D}\) equal the set of all preference orderings \(\mathcal{P}\), then \(cl(T(\mathcal{D}))\) is a convex type space, and using Saks and Yu (2005), we can immediately conclude that Theorem 1 holds. The type spaces that we will cover are not necessarily convex and this makes the results novel.
Table 2: Top Lifting Property

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$P'$</td>
<td>$P''$</td>
</tr>
<tr>
<td>$x$</td>
<td>$[a]$</td>
<td>$[a]$</td>
</tr>
<tr>
<td>$y$</td>
<td>$x$</td>
<td>$x$</td>
</tr>
<tr>
<td>$z$</td>
<td>$z$</td>
<td>$z$</td>
</tr>
<tr>
<td>$b$</td>
<td>$y$</td>
<td>$y$</td>
</tr>
<tr>
<td>$x'$</td>
<td>$y'$</td>
<td>$x'$</td>
</tr>
<tr>
<td>$y'$</td>
<td>$x'$</td>
<td>$y'$</td>
</tr>
</tbody>
</table>

We now define two notions of connectedness. We generalize the notion of $G$-neighbors we had defined earlier. Two alternatives $a$ and $b$ are **neighbors** in a domain of preference ordering $D$ if there is an ordering $P \in D$ such that $P(1) = a$ and $P(2) = b$, and another ordering $P' \in D$ such that $P'(1) = b$ and $P'(2) = a$. Now, construct an undirected graph $G(D)$ as follows. The set of nodes in $G(D)$ is the set of alternatives $A$. For any $a, b \in A$, there is an edge $\{a, b\}$ in $G(D)$ if and only if $a$ and $b$ are neighbors.

**Definition 7** A domain of preference orderings $D$ is **ordinally connected** if $G(D)$ is a connected graph, i.e., for every pair of alternatives $a, b \in A$ there is a path in $G(D)$ between $a$ and $b$.

Note that we do not require that every pair of alternatives be neighbors. We only require that the neighborhood graph is connected. Example 1 gives a domain that is ordinally connected, but in the closure of the ordinal type space induced from it, the 2-cycle monotonicity characterization result does not hold. Below, we strengthen the notion of ordinal connectedness. Our strengthening will still include the domain in Example 1. However, we will not allow for indifferences, i.e., only consider strict types. In that case, we will show that our 2-cycle monotonicity characterization holds.

We need some preliminary definitions. For any ordering $P \in D$ and for any alternative $b \equiv P(k) \neq P(1)$, an ordering $P'$ is a **local** $b$-lift of $P$ if $P'(k - 1) = P(k) = b$, $P'(k) = P(k - 1)$, and $P'(j) = P(j)$ for all $j \notin \{k, k - 1\}$. For any ordering $P$ and any alternative $b$, we say an ordering $P'$ is in the Kemeny $b$-path from $P$ if there exists a sequence of orderings $(P \equiv P^1, P^2, \ldots, P^k \equiv P')$ such that for all $j \in \{1, \ldots, k - 1\}$, $P^{j+1}$ is a local $b$-lift of $P^j$.

**Definition 8** A pair of alternatives $a, b \in A$ are **K-neighbors** if

1. for every ordering $P \in D$ with $P(1) = a$ and every $P'$ that is in the Kemeny $b$-path from $P$, we have $P' \in D$ and
2. for every ordering \( P \in \mathcal{D} \) with \( P(1) = b \) and every \( P' \) that is in the Kemeny \( a \)-path from \( P \), we have \( P' \in \mathcal{D} \).

In words, \( a \) and \( b \) are K-neighbors if whenever \( a \) is the top ranked, the preference ordering obtained by lifting \( b \) one position up belongs to the domain and whenever \( b \) is top ranked, the preference ordering obtained by lifting \( a \) one position up belongs to the domain. Note that if a domain satisfies top lifting, then for any alternative \( a \in A \), there is at least one preference ordering \( P \) such that \( P(1) = a \). Next, in such domains if \( a \) and \( b \) are K-neighbors, then they are neighbors. Now construct an undirected graph \( K(\mathcal{D}) \), where the set of nodes is \( A \) and there is an edge between a pair of alternatives \( a, b \in A \) if and only if \( a \) and \( b \) are K-neighbors.

**Definition 9** A domain of preference orderings \( \mathcal{D} \) is **K-connected** if the graph \( K(\mathcal{D}) \) is connected.

Clearly, a K-connected domain is ordinally connected. However, ordinally connected domains need not be K-connected if \( |A| \geq 4 \). The following example illustrates that.

**Example 2**

Suppose \( A = \{a, b, c, d\} \). Consider a domain \( \mathcal{D} \) consisting of the six preference orderings shown in Table 3. Clearly, this domain is ordinally connected - the graph \( G(\mathcal{D}) \) is a line graph with edges \( \{a, b\}, \{b, c\}, \{c, d\} \). But this domain is not K-connected. To see this, consider the preference ordering \( P^1 \). For \( a \) and \( d \) to be K-neighbors, we require the preference ordering \( P \) given by \( aPbPdPc \) to be in the domain. Since \( P \notin \mathcal{D} \), we get that \( a \) and \( d \) are not K-neighbors. Following a similar arguments for \( P^2 \) and \( P^4 \), we see that \( b \) and \( d \) are not K-neighbors and \( c \) and \( d \) are not K-neighbors. Hence, in the graph \( K(\mathcal{D}) \), \( d \) is not connected to any of the alternatives. Hence, this domain is not K-connected.

<table>
<thead>
<tr>
<th></th>
<th>( P^1 )</th>
<th>( P^2 )</th>
<th>( P^3 )</th>
<th>( P^4 )</th>
<th>( P^5 )</th>
<th>( P^6 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a )</td>
<td>( b )</td>
<td>( b )</td>
<td>( c )</td>
<td>( c )</td>
<td>( d )</td>
<td></td>
</tr>
<tr>
<td>( b )</td>
<td>( a )</td>
<td>( c )</td>
<td>( b )</td>
<td>( d )</td>
<td>( c )</td>
<td></td>
</tr>
<tr>
<td>( c )</td>
<td>( c )</td>
<td>( a )</td>
<td>( a )</td>
<td>( a )</td>
<td>( a )</td>
<td></td>
</tr>
<tr>
<td>( d )</td>
<td>( d )</td>
<td>( d )</td>
<td>( d )</td>
<td>( b )</td>
<td>( b )</td>
<td></td>
</tr>
</tbody>
</table>

Table 3: Connected but not K-connected

**Definition 10** A type space \( D \) is **ordinally admissible** if there is a domain of preference orderings \( \mathcal{D} \) that is K-connected and satisfies top lifting such that \( D = T(\mathcal{D}) \).
Our main result is that in ordinally admissible type spaces, 2-cycle monotonicity is necessary and sufficient for implementation. However, notice that an ordinally admissible type space consists of strict types only.

**Theorem 2** Suppose $D$ is an ordinally admissible type space. Then, $f : D \to A$ is implementable if and only if it is 2-cycle monotone.

The proof of Theorem 2 is given in the Appendix. The proof follows the exact steps that we followed to prove Theorem 1. But now with strict types, we show that our steps work in more general type spaces.

Although indifferences in ordinally admissible type spaces may break Theorem 2 (as Example 1 shows), if we assume a certain amount of continuity of the allocation rule, Theorem 2 is true even with indifferences.

**Definition 11** Let $D \equiv T(D)$ be an ordinally admissible type space. An allocation rule $f : \text{cl}(D) \to A$ satisfies condition $C^*$ if for every $t \in \text{cl}(D) \setminus D$, there exists a sequence of types $\{t^k\}_k$ in $D$ such that $\lim_{k \to \infty} t^k = t$ and $f(t) = f(t^k)$ for all $t^k$ in the sequence.

With this additional condition, Theorem 2 now extends to domains with indifferences.

**Theorem 3** Suppose $D \equiv T(D)$ is an ordinally admissible type space and $f : \text{cl}(D) \to A$ is an allocation rule satisfying condition $C^*$. Then, $f$ is implementable if and only if it is 2-cycle monotone.

**Proof:** The proof is a direct consequence of Theorem 2 and condition $C^*$. Suppose $f$ is 2-cycle monotone but not implementable. Then, it is not cycle monotone and there exists a sequence of types $t^1, \ldots, t^k$ such that

$$\sum_{j=1}^{k}[t^j(f(t^j)) - t^j(f(t^{j+1}))] < 0,$$

where $t^{k+1} \equiv t^1$. By condition $C^*$, there exists a sequence of types $s^1, \ldots, s^k$ such that for all $j \in \{1, \ldots, k\}$, $f(s^j) = f(t^j)$, $s^j \in T(D)$ and $s^j$ is arbitrarily close to $t^j$. By Theorem 2, we know that $f$ restricted to $T(D)$ satisfies cycle monotonicity. Hence,

$$\sum_{j=1}^{k}[s^j(f(t^j)) - s^j(f(t^{j+1}))] \geq 0.$$

Since $s^j$ is arbitrarily close to $t^j$, this contradicts Inequality 2. \qed

The section highlights that there are ordinal type spaces where violation of 2-cycle monotonicity can only occur at boundary - according to Theorems 2 and 3, this is the case for
any type space which is the closure of an ordinally admissible type space. The usual revenue
maximization problems (a la Myerson (1981)) seek to maximize expected payment of an
agent. Now, note that for any implementable allocation rule \( f : cl(T(D)) \rightarrow A \), we can take
its restriction in \( T(D) \) and extend it to an implementable allocation rule \( \bar{f} : cl(T(D)) \rightarrow A \)
satisfying condition \( C^* \). We know by revenue equivalence that the payment of an imple-
mentable allocation rule is uniquely determined upto a constant. Usually, this constant is
determined by the individual rationality constraints and unique for all the implementable
allocation rule if we are interested in expected revenue maximization. Hence, \( f \) and \( \bar{f} \)
must differ in payments only at boundary points. Since the boundary points have Lebesgue mea-
sure zero, the expected payment of \( f \) and \( \bar{f} \) must be the same. Hence, as far as expected
revenue maximization is concerned, condition \( C^* \) is without loss of generality.

We give two examples of \( D \) that satisfy the K-connectedness and the lifting property.
These are examples are of non-convex multidimensional type spaces, and hence, the earlier
results are silent on such examples. The single peaked type space on a tree also satisfies
K-connectedness and the lifting property, but since Theorem 1 covers it, we do not discuss
it here.

1. Semi single peaked type space. In semi single peaked type space, there is an
exogenously given ordering \( \succ \) on the set of alternatives \( A \). We say an ordering \( P \)
is single-peaked to left if for any pair of alternatives \( a, b \in A, b \succ a \) and \( a \succ P(1) \) implies
\( a \succ R b \). Similarly, an ordering \( P \) is single-peaked to right if for any pair of alternatives
\( a, b \in A, P(1) \succ a \) and \( a \succ b \) implies \( a \succ L b \). Hence, semi single peakedness requires
single peakedness to one of the sides of the peak.

The set of admissible orderings \( D \) is a semi single peaked domain if it consists of
either all left single peaked orderings or all right single peaked orderings. Consider
a semi single peaked domain \( D \) and assume that it consists of all right single peaked
orderings. Then, alternatives \( a \) and \( b \) are neighbors if \( B(a, b) = \emptyset \), where \( B(a, b) \) is the
set of alternatives between \( a \) and \( b \) according to \( \succ \). Note that \( a \) can be top ranked
in an ordering and any alternative \( b \in L(a) \) can be second ranked. However, if \( b \) is
top ranked, \( a \) can be second ranked only if \( B(a, b) = \emptyset \). For this reason, \( G(D) \) is a
line graph, which is connected. It is also easy to see that every neighbor of \( a \) is also a
K-neighbor. This is because if \( b \) is a neighbor of \( a \), then it can be lifted to any rank
from a given preference ordering. Hence, \( K(D) \) is also connected.

We can verify that the semi single peaked domain satisfies top lifting. To see this,
consider an ordering \( P \in D \), where \( a \succ R b \). If \( b \in L(a) \), we can construct an ordering
where \( a \) is top ranked and \( b \) is second ranked by lowering all the alternatives (except \( a \))
below \( b \) but maintaining single peaked to the right of \( a \). If \( b \in R(a) \), then alternatives
to the left of $a$ can be lowered sufficiently to make $a$ the peak and it will automatically maintain single peakedness to the right of $a$.

The interior of the type space of Example 1 is semi single peaked. To see this, consider the exogenous ordering $a \succ b \succ c$. The set of all right single peaked preference orderings with respect to $\succ$ is exactly the preference orderings shown in Example 1. As we had shown, Theorem 1 does not apply to this type space but Theorem 2 applies.

2. Single peaked type space with characteristics. This is a generalization of the single peaked type space. We are now exogenously given a set of orderings $\mathcal{S}$ over the set of alternatives. The domain $\mathcal{D}$ consists of all orderings that are single peaked with respect to some $\succ \in \mathcal{S}$. If $\mathcal{S}$ is a singleton, this is precisely the single peaked domain. Suppose the set of alternatives are objects. An element of $\mathcal{S}$ can be interpreted as a "characteristic" of the objects. Depending on the characteristic used by an agent to rank the objects, his preference must be single peaked with respect to that characteristic.

Consider an example with $A = \{a, b, c, x, y\}$ and let $\mathcal{S} = \{\succ_1, \succ_2\}$, where $a \succ_1 b \succ_1 c \succ_1 x \succ_1 y$ and $y \succ_2 a \succ_2 b \succ_2 x \succ_2 c$. Figure 1 shows the graph $G(\mathcal{D})$ for this domain. The edges are derived from the single peaked restrictions on each characteristic.

![Graph K(D) for the single peaked domain with two characteristics.](image)

In general, the graph $K(\mathcal{D})$ is connected since $\mathcal{D}$ contains the single peaked domain, which is connected. Also, $\mathcal{D}$ satisfies the top lifting property. To see this, consider any $a, b \in A$ and suppose there is a preference ordering $P$ where $aPb$. Since $P$ is single peaked with respect to some $\succ \in \mathcal{S}$, we can apply the arguments for the single peaked domain to show that there is some ordering $P'$ that is single peaked with respect to $\succ$ such that top lifting holds for $a$ and $b$ at $P$. 

21
5 Domains with Free Triple at the Top

So far, we have discussed domains that are K-connected (the single peaked domain on a tree in Theorem 1 is also a K-connected domain). In this section, we identify an ordinal domain that is not K-connected and still every 2-cycle monotone allocation rule is implementable in the type space induced by this domain. A primary objective of this section is to illustrate the fact that even though the assumptions of Theorems 1 and 2 may not be satisfied in some type spaces, we can still employ our methodology of proof to derive an analogue of these results.

Our domain uses the following definition. We say alternatives $a, b, c \in A$ are a free triple at the top in domain $D$ if for every $x, y, z \in \{a, b, c\}$ with $x \neq y \neq z$, there exists $P \in D$ such that $P(1) = x, P(2) = y, P(3) = z$. In other words, there exists six distinct orderings where $a, b, c$ occupy top three ranks.

**Definition 12** A domain $D$ satisfies **free triple at the top (FTT)** if every three distinct alternatives in $A$ are a free triple at the top in $D$.

We make some observations about FTT domains. First, an FTT domain is ordinally connected since any pair of alternatives can be ranked first and second. However, it need not be K-connected. To see this consider the following example with $A = \{a, b, c, d, e\}$. Suppose whenever $a, b, c$ occupy the top 3 positions, $d$ is better than $e$ and whenever $a, b, d$ occupy top 3 positions $c$ is better than $e$. It is easy to construct an FTT domain that satisfies this restriction. A consequence of these restrictions is that there are preference orderings where one of $\{a, b, c, d\}$ is top ranked and $e$ is fifth ranked but local $e$-lift is not possible (i.e., $e$ cannot become fourth ranked) from these preference orderings. Then, it is clear that $a, b, c, d$ are not K-neighbors of $e$. As a result, in the graph $K(D)$, $e$ is not connected to any alternative.

We are now ready to state the main result of this section.

**Theorem 4** Suppose $D = T(D)$ or $D = cl(T(D))$, where $D$ is an FTT domain. Then, $f : D \rightarrow A$ is implementable if and only if it is 2-cycle monotone.

The proof is given in the appendix. After an initial step, the proof uses the general methodology developed in the proof of Theorem 1. We give two examples where the underlying domain is an FTT domain.

---

9If we did not focus on ordinal type spaces, then a type space not satisfying K-connectedness but where 2-cycle monotonicity implies implementability is easy to find - this follows from the fact in any convex type space, 2-cycle monotonicity implies implementability.
Individual versus social ranking. Suppose the alternatives are public projects. There is a “social ranking” \( \succ \) of these projects. The type of an agent consists of some component of his private preference and the remaining of the social ranking. In particular, the agent ranks any three projects as his top three and then follows the social ranking for the remaining projects. Such a domain will satisfy the FTT assumption.

Complete type space. In the complete type space, \( \mathcal{D} \) is the set of all possible orderings over \( A \). Clearly, this is an FTT domain. Note that the complete type space covers the multi-object auction model with unit demand. To see this, let \( A \) be the set of heterogeneous objects and agent \( i \) can be assigned exactly one object from \( A \). Then, the complete type space assumption requires that any vector of non-negative valuations can be assigned to the objects. However, the complete type space is convex, and, hence, Theorem 4 is already implied by the existing results in the literature for such a type space.

6 Payments and Revenue Equivalence

It is well know that if \( f \) is implementable, then the following payment rule implements \( f \). Fix a type \( s \in D \) and set \( p(t) = 0 \) for all \( t \) with \( f(t) = f(s) \). For all \( t \in D \) such that \( f(t) \neq f(s) \), set \( p(t) \) equal to \( \text{dist}(f(s), f(t)) \). If \( f \) is cyclically monotone, then, \( p \) implements \( f \) - see for instance, Vohra (2011) and Kos and Messner (2013).

The characterization of the set of all payment rules that implement an allocation rule is done using the revenue equivalence principle.

Definition 13 An allocation rule \( f \) satisfies revenue equivalence if for all payment rules \( p, q \) that implement \( f \), there exists a constant \( \alpha \in \mathbb{R} \) such that for all \( t \in D \)

\[
p(t) = q(t) + \alpha.
\]

The revenue equivalence holds in ordinal type spaces under weaker conditions than the conditions we have discussed for the 2-cycle monotonicity characterization. We are going to show revenue equivalence in ordinally connected type spaces. To remind, a set of preference orderings \( \mathcal{D} \) is ordinally connected, if the graph \( G(\mathcal{D}) \) is connected.

Theorem 5 Suppose \( \mathcal{D} \) is ordinally connected and \( D = T(\mathcal{D}) \) or \( D = \text{cl}(T(\mathcal{D})) \). Then every implementable allocation rule \( f : D \rightarrow A \) satisfies revenue equivalence.

The proof of Theorem 5 is in the Appendix. The proof essentially follows by showing a counterpart of Lemma 8 (see Appendix) for ordinally connected type spaces and then
using standard results from the literature. We remark that Chung and Olszewski (2007) and Heydenreich et al. (2009) have shown that if $D$ is a topologically connected subset of $\mathbb{R}^{|A|}$, then every implementable allocation rule satisfies revenue equivalence in such a type space. However, since $D$ consists of strict orderings, $T(D)$ is not topologically connected and hence, our result is not a direct corollary of their results. However, both these papers provide sufficient conditions using which revenue equivalence can be checked in our ordinal type spaces. The proof uses such conditions.

In the classical literature on multidimensional mechanism design a type space is usually a subset of $\mathbb{R}^{|A|}$ possessing some geometrical properties. The state-of-art in that literature is that if the type space is a topologically connected subset of $\mathbb{R}^{|A|}$, then every implementable allocation rule satisfies revenue equivalence (Chung and Olszewski, 2007; Heydenreich et al., 2009). On the other hand, if the type space is convex, then every 2-cycle monotone allocation rule is implementable. Some parallel between our results and these results can be drawn as follows. Theorem 5 shows that in ordinally connected type spaces, every implementable allocation rule satisfies revenue equivalence. On the other hand, Theorem 2 shows that in ordinally admissible type spaces (which requires K-connectedness and top lifting), every 2-cycle monotone allocation rule is implementable.

An ordinally connected type space allows us to be precise on the nature of the shortest paths between any pair of nodes in $G^f$. Suppose $f$ is implementable. Now, for any pair of alternatives $a, b \in A$, consider any path $(a_1, \ldots, a_k)$ in $G(D)$, where $a_1 \equiv a$ and $b \equiv a_k$. Then, $dist^f(a, b) = \sum_{j=1}^{k-1} d(a_j, a_{j+1})$. This follows from the fact that

$$0 = dist^f(a, b) + dist^f(b, a)$$

$$\leq \sum_{j=1}^{k-1} d(a_j, a_{j+1}) + \sum_{j=k-1}^{1} d(a_{j+1}, a_j)$$

$$= \sum_{j=1}^{k-1} [d(a_j, a_{j+1}) + d(a_{j+1}, a_j)]$$

$$= 0,$$

where the first equality follows from Theorem 5 and the last equality from Lemma 2. Since in many examples, we know the structure of $G(D)$, this allows us to know the payments in these type spaces explicitly. Moreover, it can be shown that these particular payments occupy a central role among the set of all payments - Kos and Messner (2013) contain a detailed discussion on this topic in very general type spaces.
We now discuss the literature and its relation to our results. As discussed earlier, in the one-dimensional model of single object auctions, Myerson (1981) characterizes implementable allocation rules using a monotonicity condition, which is equivalent to 2-cycle monotonicity - see also Spence (1974); Mirrlees (1976). The cycle monotonicity characterization in Rochet (1987) can be thought of as an extension of Myerson’s characterization to multidimensional models. The recent literature on multidimensional mechanism design started with the paper of Jehiel et al. (1999) who observed that besides 2-cycle monotonicity, an integral condition is required to ensure Bayesian implementability in multidimensional environments with randomization. However, if the set of alternatives is finite, the allocation rule is deterministic and the type space is convex, only 2-cycle monotonicity is sufficient (Bikhchandani et al., 2006; Saks and Yu, 2005; Ashlagi et al., 2010; Gui et al., 2004; Cuff et al., 2012). Our results are extensions of these results to non-convex type spaces. Mishra and Roy (2013) also consider a non-convex type space, which they call rich dichotomous type space, and show that 3-cycle monotonicity is sufficient for implementability in their type space but 2-cycle monotonicity is not sufficient.

A parallel literature in multidimensional mechanism design pursues type spaces where revenue equivalence result in Myerson (1981) holds. Contributions to this are Krishna and Maenner (2001); Milgrom and Segal (2002); Chung and Olszewski (2007); Heydenreich et al. (2009); Carbajal (2010); Kos and Messner (2013). We use a characterization in Heydenreich et al. (2009) to prove revenue equivalence in a large class of ordinal type spaces.

Most of the type space restrictions in multidimensional mechanism design is geometric (using assumptions like convexity or connectedness in topological spaces). Our ordinally admissible domain formulation is influenced by a vast literature in strategic social choice theory where transfers are not allowed. For instance, the connectedness and lifting properties we discuss have close resemblance to similar properties being used to identify dictatorial domains (Aswal et al., 2003), median domains (Chatterji et al., 2013; Nehring and Puppe, 2007), tops-only domains (Chatterji and Sen, 2011; Weymark, 2008) in social choice theory. We find it interesting to observe that such conditions could be used in multidimensional mechanism design models with transfers to derive sufficient conditions for implementability. Since most of our non-convex type spaces are single peaked type spaces or their generalizations, we will like to point out that strategic social choice theory, starting with Moulin (1980)

\[ 10 \] There are many papers which characterize different extensions of implementability in convex type spaces using 2-cycle monotonicity and additional technical conditions - for Bayes-Nash implementation, see Jehiel et al. (1999) and Muller et al. (2007); for randomized implementation, see Archer and Kleinberg (2008); for implementation with general value functions, see Berger et al. (2010) and Carbajal and Ely (2013); for extension of cycle monotonicity to general environments, see Rahman (2011).
and Sprumont (1991), have a long tradition of studying these type spaces without monetary transfers. However, allowing for transfers in many of these type spaces is practical in many of these models. Hence, our results extend this literature to the case of transfers.

The idea of imposing ordinal restriction on type spaces is a relatively new idea in this literature. We are aware of only one instance where such an idea of ordinal restriction was pursued in this literature. The order-based type space considered in Bikhchandani et al. (2006) is defined by considering a weak partial order on the set of alternatives and every type must induce this order. Firstly, an order-based type space is convex. Second, we consider a set of strict and complete orders and a type in our type space must induce one of the orders in this set. In that sense, our type space restrictions are different from the order-based type space, and neither is stronger than the other.

We conclude by giving a tabular representation of the literature and where our results fit in. Table 4 describes type spaces discussed in the literature and in the current paper, where 2-cycle monotonicity characterization and revenue equivalence result holds.

<table>
<thead>
<tr>
<th>Geometric Restrictions on Type Spaces</th>
<th>Revenue Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Cycle Monotonicity Characterization</td>
<td>Connectedness</td>
</tr>
<tr>
<td>Convexity</td>
<td>(Chung and Olszewski, 2007)</td>
</tr>
<tr>
<td>(Saks and Yu, 2005)</td>
<td></td>
</tr>
<tr>
<td>(Ashlagi et al., 2010)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ordinal Restriction on Type Spaces</th>
<th>Revenue Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>2-Cycle Monotonicity Characterization</td>
<td>Ordinal connectedness (this paper)</td>
</tr>
<tr>
<td>Order-based type space</td>
<td></td>
</tr>
<tr>
<td>(Bikhchandani et al., 2006)</td>
<td></td>
</tr>
<tr>
<td>Single peaked type space on a tree,</td>
<td></td>
</tr>
<tr>
<td>Ordinally admissible strict type space,</td>
<td></td>
</tr>
<tr>
<td>Free triple at the top type space (this paper)</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: Literature and our results

**APPENDIX: OMITTED PROOFS**

**Proof of Theorem 2**

We follow the steps in Theorem 1. Obviously, some of the steps of Theorem 1 continues to apply in this type space. So, we only do the steps that do not apply. Lemma 1 continues to hold. The first step of the proof of Theorem 2 is the analogue of Lemma 2.
Lemma 8  If $a, b$ are neighbors, then $d(a, b) + d(b, a) = 0$. \footnote{Note that we do not require $a$ and $b$ to be K-neighbors.}

Proof:  Consider $a, b \in A$ such that $a$ and $b$ are neighbors. By Lemma 1, $d(a, b) + d(b, a) \geq 0$. Assume for contradiction $d(a, b) + d(b, a) = \epsilon > 0$. Then, either $d(a, b) > \frac{\epsilon}{2}$ or $d(b, a) > \frac{\epsilon}{2}$. Suppose $d(a, b) > \frac{\epsilon}{2}$ - a similar proof works if $d(b, a) > \frac{\epsilon}{2}$. Then, there is a type $s \in D(b)$ such that $d(a, b) \leq s(b) - s(a) < d(a, b) + \epsilon_1$, for any $\epsilon_1 > 0$ arbitrarily close to zero, in particular $\epsilon_1 < \frac{\epsilon}{2}$. Hence, $s(b) - s(a) > \frac{\epsilon}{2}$. We now choose a $\delta \in (2\epsilon_1, s(b) - s(a))$ but arbitrarily close to $2\epsilon_1$. Since $a$ and $b$ are neighbors, there exists a $P \in D$ such that $b$ is top ranked and $a$ is second ranked. We can construct a type $u \in D$ that induces $P$ and

$$u(x) = \begin{cases} 
    s(x) + \delta & \text{if } x = a \\
    s(x) + \frac{\delta}{2} & \text{if } x = b \\
    \min(s(x), s(a)) & \text{if } x \notin \{a, b\}, 
\end{cases}$$

Notice that since $s(b) > s(a)$, we have $u(b) > u(a)$ for sufficiently small $\delta > 2\epsilon_1$. Also, alternatives other than $a$ and $b$ are ordered according to $P$ but their values are not increased.

We will now argue that $f(u) = a$. First, if $f(u) = x \notin \{a, b\}$, we have $u(x) - u(b) \leq s(x) - s(b) - \frac{\delta}{2} < s(x) - s(b)$, which violates 2-cycle monotonicity. Second, if $f(u) = b$, we have $u(b) - u(a) = s(b) - s(a) - \frac{\delta}{2} < d(a, b) - (\frac{\delta}{2} - \epsilon_1) < d(a, b)$, which violates the definition of $d(a, b)$. Hence, $f(u) = a$.

But this implies that $d(b, a) \leq u(a) - u(b) = s(a) - s(b) + \frac{\delta}{2} \leq -d(a, b) + \frac{\delta}{2}$. Hence, $d(b, a) + d(a, b) \leq \frac{\delta}{2}$. Since $\delta, \epsilon_1$ can be chosen arbitrarily close to zero, this contradicts the fact that $d(a, b) + d(b, a) = \epsilon > 0$. 

The next lemma establishes the counterpart of Lemma 3. For any pair of alternatives, $a, c$ we will consider paths in $K(D)$ between $a$ and $c$. Since we assume $D$ to be K-connected, there is at least one path between $a$ and $c$. A path between $a$ and $c$ is direct if it involves only $a$ and $c$. A path between $a$ and $c$ is indirect if it is not direct. By definition, there is a direct path between $a$ and $c$ if and only if $a$ and $c$ are K-neighbors.

Lemma 9  For every pair of alternatives $a, c \in A$ and any indirect path $\Pi(a, c)$ between $a$ and $c$ in $K(D)$, there exists an alternative $b$ in this path such that $d(a, b) + d(b, c) \leq d(a, c)$.

Proof:  Fix $a, c \in A$ and an indirect path $\Pi(a, c)$ between $a$ and $c$ in $K(D)$. Choose an $\epsilon > 0$ arbitrarily close to zero and a $t \in D(c)$ such that $d(a, c) \leq t(c) - t(a) < d(a, c) + \epsilon$. We consider two cases.

Case 1. $t(c) > t(a)$. Choose an alternative $b$ in $\Pi(a, c)$ such that $b$ is a K-neighbor of $c$. Let the ordering induced by $t$ be $P$. By top lifting, there exists an ordering $P' \in D$ such that (a)
Let the ordering induced by $d$ to zero, we conclude that $x \neq c$ and
$b$ is the neighbor of $c$ in $\Pi(c,a)$, by K-connectedness, the ordering induced by $s$ belongs to $D$. Hence, $s \in D$.

We argue that $f(s) = b$. First, suppose $f(s) = x \notin \{b, c\}$. Then, $s(x) - s(c) < t'(x) - t'(c)$, and this contradicts 2-cycle monotonicity. Next, suppose $f(s) = c$. Then, $d(b,c) \leq s(c) - s(b) = t'(c) - t'(b) - \delta < d(b,c)$, a contradiction. Hence, $f(s) = b$.

Now, $d(a,b) \leq s(b) - s(a) = |t'(b) - t'(a)| + \delta + \bar{\varepsilon}$. Since $\delta = |t'(c) - t'(b)| - d(b,c) + \varepsilon''$, we have $d(a,b) \leq |t'(c) - t'(a)| - d(b,c) + \bar{\varepsilon} + \varepsilon'' < d(a,c) + \varepsilon - d(b,c) + \bar{\varepsilon} + \varepsilon''$. This implies that $d(a,b) + d(b,c) < d(a,c) + \varepsilon + \bar{\varepsilon} + \varepsilon''$. Since $\varepsilon, \bar{\varepsilon}$ and $\varepsilon''$ can be chosen arbitrarily close to zero, we conclude that $d(a,b) + d(b,c) \leq d(a,c)$.

**Case 2.** $t(c) < t(a)$. Choose an alternative $b$ in $\Pi(a,c)$ such that $b$ is a K-neighbor of $a$. Let the ordering induced by $t$ be $P$. By top lifting, there exists an ordering $P' \in D$ such that (a) $P'(1) = a$ and (b) if $cP'c'$ for any alternative $c'$ then $cP'c'$. Hence, we can construct a type $t' \in D$ that induce $P'$ and $t'(c) = t(c) + \varepsilon'$, $t'(a) = t(a)$, and $t'(x) \leq t(x)$ for all $x \notin \{a,c\}$, where $\varepsilon' > 0$ but arbitrarily close to zero, in particular, $\varepsilon' < \varepsilon$. Since $t'(c) > t(c)$ and $t'(x) \leq t(x)$ for all $x \neq c$, we have $|t'(x) - t'(c)| + |t(c) - t(x)| < 0$ for all $x \neq c$, and hence, by 2-cycle monotonicity, $f(t') = c$. Further, $d(a,c) \leq t'(c) - t'(a) < d(a,c) + \varepsilon$.

Let $\delta = t'(c) - t'(b) - d(b,c) + \varepsilon''$, for some $\varepsilon'' > 0$ but arbitrarily close to zero. Since $f(t') = c$, we have $t'(c) - t'(b) \geq d(b,c)$. Hence, $\delta > 0$ but arbitrarily close to $t'(c) - t'(b) - d(b,c)$. Now, we construct a new type $s$ as follows. Choose an $\bar{\varepsilon} > 0$ but arbitrarily close to zero.

$$s(x) = \begin{cases} 
  t'(x) + \bar{\varepsilon} & \text{if } x = c \\
  t'(x) + \delta + \bar{\varepsilon} & \text{if } x = b \\
  t'(x) & \text{if } x \in A \setminus \{b,c\}
\end{cases}$$

Since $a$ is top at $t'$ and $b$ is the neighbor of $a$ in $\Pi(c,a)$, by K-connectedness, the ordering induced by $s$ belongs to $D$ for small enough $\bar{\varepsilon}$. Hence, $s \in D$.  

$P'(1) = c$ and (b) if $aP'c'$ for any alternative $c'$ then $aP'c'$. Hence, we can construct a type $t' \in D$ that induce $P'$ and $t'(c) = t(c) + \varepsilon$, $t'(a) = t(a)$, and $t'(x) \leq t(x)$ for all $x \notin \{a,c\}$, where $\varepsilon > 0$ but arbitrarily close to zero such that $\varepsilon' < \varepsilon$. Since $t'(c) > t(c)$ and $t'(x) \leq t(x)$ for all $x \neq c$, we have $|t'(x) - t'(c)| + |t(c) - t(x)| < 0$ for all $x \neq c$, and hence, by 2-cycle monotonicity, $f(t') = c$. Further, $d(a,c) \leq t'(c) - t'(a) < d(a,c) + \varepsilon$. 

Let $\delta = t'(c) - t'(b) - d(b,c) + \varepsilon''$, for some $\varepsilon'' > 0$ but arbitrarily close to zero. Since $f(t') = c$, we have $t'(c) - t'(b) \geq d(b,c)$. Hence, $\delta > 0$ but arbitrarily close to $t'(c) - t'(b) - d(b,c)$.
We argue that \( f(s) = b \). First, suppose \( f(s) = x \notin \{b, c\} \). Then, \( s(x) - s(c) < t'(x) - t'(c) \), and this contradicts 2-cycle monotonicity. Next, suppose \( f(s) = c \). Then, \( d(b, c) \leq s(c) - s(b) = t'(c) - t'(b) - \delta < d(b, c) \), a contradiction. Hence, \( f(s) = b \).

Now, \( d(a, b) \leq s(b) - s(a) = [t'(b) - t'(a) + \delta] + \epsilon \). Since \( \delta = [t'(c) - t'(b)] - d(b, c) + \varepsilon'' \), we have \( d(a, b) \leq [t'(c) - t'(a)] - d(b, c) + \varepsilon + \varepsilon'' < d(a, c) + \varepsilon - d(b, c) + \varepsilon + \varepsilon'' \). This implies that \( d(a, b) + d(b, c) < d(a, c) + \varepsilon + \varepsilon'' \). Since \( \epsilon, \varepsilon \) and \( \varepsilon'' \) can be chosen arbitrarily close to zero, we conclude that \( d(a, b) + d(b, c) \leq d(a, c) \).

Once we have established these major lemmas. The remaining Lemmas in the proof of Theorem 1 goes through. We state their generalizations and skip the proof since it mirrors the proof given in their counterparts for the proof of Theorem 1.

**Lemma 10** For any pair of alternatives \( a_1, a_k \in A \), let \((a_1, a_2, \ldots, a_k)\) be an indirect path \( \Pi(a_1, a_k) \) between \( a_1 \) and \( a_k \) in \( K(D) \). Then, the following are true.

\[
\begin{align*}
 & d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{k-1}, a_k) \leq d(a_1, a_k) \\
 & d(a_k, a_{k-1}) + d(a_{k-1}, a_{k-2}) + \ldots + d(a_2, a_1) \leq d(a_k, a_1).
\end{align*}
\]

**Lemma 11** Suppose for every sequence of alternatives \((a_1, \ldots, a_k)\), we have

\[
\sum_{j=1}^{k} d(a_j, a_{j+1}) \geq 0,
\]

where \( a_{k+1} \equiv a_1 \). Then, \( f \) is cyclically monotone.

**Lemma 12** Suppose \((a_1, \ldots, a_k)\) is a path in \( K(D) \). Then,

\[
\sum_{j=1}^{k} d(a_j, a_{j+1}) \geq 0,
\]

where \( a_{k+1} \equiv a_1 \).

**Proof:** If \( k = 2 \), then the claim is true by 2-cycle monotonicity. Else, \( k > 2 \) and \( a_j \) and \( a_{j+1} \) are \( G \)-neighbors for all \( j \in \{1, \ldots, k-1\} \). By Lemma 10, \( d(a_k, a_1) \geq d(a_k, a_{k-1}) + \ldots + d(a_2, a_1) \). Hence,

\[
\begin{align*}
 d(a_1, a_2) + d(a_2, a_3) + \ldots + d(a_{k-1}, a_k) + d(a_k, a_1) & \geq \sum_{j=1}^{k-1} [d(a_j, a_{j+1}) + d(a_{j+1}, a_j)] \\
 & = 0,
\end{align*}
\]

29
where the last equality follows from the fact that \( a_j \) and \( a_{j+1} \) are \( G \)-neighbors for all \( j \in \{1, \ldots, k - 1\} \) and Lemma 8.

At this point, it will be useful to consider another graph \( G^f \). The set of nodes in \( G^f \) is the set of alternatives \( A \). It is a complete directed graph. Hence, for every pair of alternatives \( a, b \in A \), there is an edge from \( a \) to \( b \) and an edge from \( b \) to \( a \). The path from an alternative \( a \) to another alternative \( b \) in \( G^f \) is a directed path. Note that for every path \((a_1, a_2, \ldots , a_k)\) in \( G^f \) from \( a_1 \) to \( a_k \), the corresponding undirected path may or may not exist in \( K(D) \). For any pair of alternatives \( a_1, a_k \in A \), denote by \( \text{dist}^f(a_1, a_k) \) the shortest path length from \( a_1 \) to \( a_k \) in \( G^f \).

**Lemma 13** For any pair of alternatives \( a, b \in A \), there exists a path \((a_1, a_2, \ldots , a_k)\) in \( K(D) \), where \( a \equiv a_1 \) and \( b \equiv a_k \), such that

\[
\sum_{j=1}^{k-1} d(a_j, a_{j+1}) = \text{dist}^f(a, b).
\]

**Proof:** Fix \( a, b \in A \) and choose a shortest path from \( a \) to \( b \) in \( G^f \). Note that \( G^f \) consists of directed edges and, hence, this path consists of directed edges. Let this path be \((a'_1, \ldots , a'_h)\), where \( a'_1 \equiv a \) and \( a'_h \equiv b \). Also, there is a unique path \( \Pi(a, b) \) between \( a \) and \( b \) in \( G \), and this consists of undirected edges in \( G \). We will sometimes refer to this path in \( G^f \) by turning them into directed edges - this will give rise to two directed paths, one from \( a \) to \( b \) and the other from \( b \) to \( a \). We will denote these two directed paths corresponding to \( \Pi(a, b) \) in \( G^f \) as \( \bar{\Pi}(a, b) \) and \( \bar{\Pi}(b, a) \).

Now, take any (directed) edge \((x, y)\) in the path \((a'_1, \ldots , a'_h)\). If \( x \) and \( y \) are not \( G \)-neighbors, then we can pick the path \( \bar{\Pi}(x, y) \equiv (x, c_1, \ldots , c_r, y) \) from \( x \) to \( y \), and \( d(x, y) \geq d(x, c_1) + d(c_1, c_2) + \ldots + d(c_{r-1}, c_r) + d(c_r, y) \) by Lemma 4. Combining such paths \( \bar{\Pi}(a'_j, a'_{j+1}) \) for all \( j \in \{1, \ldots , h - 1\} \), we get the path \( \bar{\Pi}(a, b) \) from \( a \) to \( b \) in \( G \), which we denote by \((a_1, \ldots , a_k)\) with \( a \equiv a_1 \) and \( b \equiv a_k \), and some cycles in \( G^f \). Since these cycles all consist of edges from \( G \), by Lemma 12, these cycles have non-negative length (according to weights defined in \( G^f \)). Hence, \( \text{dist}^f(a, b) \geq \sum_{j=1}^{k-1} d(a_j, a_{j+1}) \). By definition, \( \text{dist}^f(a, b) \leq \sum_{j=1}^{k-1} d(a_j, a_{j+1}) \). Hence, \( \text{dist}^f(a, b) = \sum_{j=1}^{k-1} d(a_j, a_{j+1}) \). ■

Again, the proof of Lemma 13 mirrors the proof of Lemma 6 and is skipped. This leads to the final lemma in the proof of Theorem 2.

**Lemma 14** Every cycle of \( G^f \) has non-negative length.

\(^{12}\)In Heydenreich et al. (2009), this graph is called the *allocation graph.*
The proof of Lemma 14 is similar to the proof of Lemma 7 and is skipped. Lemmas 14 and 11 establish that \( f \) is cyclically monotone, and hence, implementable. This completes the proof of Theorem 2.

Proof of Theorem 5

We start by observing that though we proved Lemma 2 by assuming the type space to be \( T(\mathcal{D}) \), where \( \mathcal{D} \) is ordinally connected, the same proof goes through even if we assume the type space to be \( cl(T(\mathcal{D})) \).

Now, the remainder of the proof can be easily done using existing results. Heydenreich et al. (2009) showed that an implementable allocation rule \( f \) satisfies revenue equivalence if and only if \( dist^f(a, b) + dist^f(b, a) = 0 \) for all \( a, b \in A \). We show that this property is satisfied in our ordinally connected domains. To see this, fix a pair of alternatives, \( a, b \in A \). Since \( f \) is cyclically monotone, \( dist^f(a, b) + dist^f(b, a) \geq 0 \) - the union of a shortest path from \( a \) to \( b \) and a shortest path from \( b \) to \( a \) gives rise to cycles, which have non-negative length due to cycle monotonicity.

But take any path in \( G(\mathcal{D}) \) from \( a \) to \( b \) (since the type space is ordinally connected, such a path will always exist). Let this path be \((a \equiv a_0, a_1, \ldots, a_k \equiv a)\). Now, consider the path \((b, a_k, \ldots, a_1, a_0 \equiv a)\). The sum of lengths of these paths is \( \sum_{j=0}^{k-1} d(a_j, a_{j+1}) = 0 \), where the equality followed from Lemma 2. Since \( dist^f(a, b) + dist^f(b, a) \) is less than or equal to the sum of lengths of these paths, and we already know that \( dist^f(a, b) + dist^f(b, a) \geq 0 \), we conclude that \( dist^f(a, b) + dist^f(b, a) = 0 \).

Proof of Theorem 4

We already know that if \( f \) is implementable, then it is 2-cycle monotone. So, we only show that if \( f \) is 2-cycle monotone, then it is implementable. We do the proof in two steps. As in the proof of Theorem 5, we note here that Lemma 2 holds in a type space of the form \( T(\mathcal{D}) \) or \( cl(T(\mathcal{D})) \) as long as \( \mathcal{D} \) is ordinally connected.

Step 1. We say a domain \( \mathcal{D} \) satisfies free pair at the top (FPT) if every pair of alternatives is a neighbor in \( \mathcal{D} \) (i.e., they can be ranked first and second at some ordering in \( \mathcal{D} \)). Clearly, a domain that satisfies FTT also satisfies FPT. Further, an FPT domain is ordinally connected. Hence, if \( \mathcal{D} \) satisfies FPT, then for every \( a, b \in A \), \( d(a, b) + d(b, a) = 0 \).

Now, consider a sequence of alternatives \((a_1, \ldots, a_k)\). By Lemma 5, if we show

\[
\sum_{j=1}^{k} d(a_j, a_{j+1}) \geq 0,
\]
where $a_{k+1} \equiv a_1$, then $f$ is implementable. We show this using induction on $k$.

In this step, we show that if $f$ is 3-cycle monotone, then it is cyclically monotone. If $k = 2$ or $k = 3$, then we are done using 3-cycle monotonicity. If $k > 3$, we pick any $j' \in \{1, \ldots, k-2\}$ and choose alternatives $a_{j'}, a_{j'+2}$. Note that $d(a_{j'}, a_{j'+2}) + d(a_{j'+2}, a_{j'}) = 0$. Now, the big cycle can be broken into sum of two smaller cycles using the pair of edges $(a_{j'}, a_{j'+2})$ and $(a_{j'+2}, a_{j'})$ as follows:

$$
\sum_{j=1}^{k} d(a_j, a_{j+1}) = \left[ \sum_{j=1}^{j'-1} d(a_j, a_{j+1}) + d(a_{j'}, a_{j'+2}) + \sum_{j=j'+2}^{k} d(a_j, a_{j+1}) \right] \\
+ \left[ d(a_{j'}, a_{j'+1}) + d(a_{j'+1}, a_{j'+2}) + d(a_{j'+2}, a_{j'}) \right]
$$

where the last inequality followed from the fact that the cycles $(a_1, \ldots, a_j, a_{j'+2}, \ldots, a_k, a_1)$ and $(a_{j'}, a_{j'+1}, a_{j'+2}, a_{j'})$ has less than $k$ nodes and our induction hypothesis applies.

**Step 2.** In this step, we show that in an FTT domain, every 2-cycle monotone allocation rule is 3-cycle monotone. Consider any triple of alternatives $a, b, c \in A$. We need to show that $d(a, b) + d(b, c) + d(c, a) \geq 0$. If $\min(d(a, b), d(b, c), d(c, a)) \geq 0$, then we are done. Suppose, without loss of generality, $d(c, a) < 0$. Using the fact that $d(x, y) + d(y, x) = 0$ for all $x, y \in A$ (due to FTT and Lemma 2), it is sufficient to show that $d(c, b) + d(b, a) \leq d(c, a)$.

To show this, we consider a type $t$ such that $f(t) = a$ and $d(c, a) \leq t(a) - t(c) < d(c, a) + \epsilon$ for some $\epsilon > 0$ but arbitrarily close to zero. Hence, $t(a) < t(c)$. Since FTT is satisfied, there exists a preference ordering $P$ such that $P(1) = c, P(2) = a$, and $P(3) = b$. Further, we can construct a type $s$ that induces $P$ as follows: $s(a) = t(a) + \epsilon_1, s(c) = t(c)$, and $s(x) < t(x)$ for all $x \notin \{a, c\}$, where $\epsilon_1 > 0$ and arbitrarily close to zero but in particular much smaller than $\epsilon$. By 2-cycle monotonicity, $f(s) = a$ - if $f(s) = y \neq a$, then $s(y) - s(a) < t(y) - t(a)$, violating 2-cycle monotonicity. Also, $\epsilon_1$ can be chosen sufficiently small such that $d(c, a) \leq s(a) - s(c) < d(c, a) + \epsilon$.

Now, let $\delta = s(a) - s(b) - d(b, a)$. Note that since $s(a) - s(b) \geq d(b, a)$, we have $\delta \geq 0$. We construct a type $\hat{s}$ such that $\hat{s}(b) = s(b) + \delta + 2\epsilon'$, $\hat{s}(a) = s(a) + \epsilon'$, and $\hat{s}(x) = s(x)$ for all $x \notin \{a, b\}$, where $\epsilon' > 0$ but arbitrarily close to zero. We argue that $f(\hat{s}) = b$. This is because if $f(\hat{s}) = x \notin \{a, b\}$, then $\hat{s}(x) - \hat{s}(a) < s(x) - s(a)$, contradicting 2-cycle monotonicity. Next, if $f(\hat{s}) = a$, then $d(b, a) \leq \hat{s}(a) - \hat{s}(b) = s(a) - s(b) - \delta - \epsilon' < d(b, a)$, a contradiction.

Hence, $f(\hat{s}) = b$. This implies that $d(c, b) \leq \hat{s}(b) - \hat{s}(c) = s(b) - s(c) + \delta + 2\epsilon' = s(a) - s(c) - d(b, a) + 2\epsilon' < d(c, a) - d(b, a) + \epsilon + 2\epsilon'$. Since $\epsilon$ and $\epsilon'$ can be chosen arbitrarily close to zero, we get $d(c, b) + d(b, a) \leq d(c, a)$. This completes the proof.
References


