A Model of Persuasion with a Boundedly Rational Agent

Jacob Glazer
Tel Aviv University and
Boston University

and

Ariel Rubinstein
Tel Aviv University and
New York University

Abstract

The paper introduces a mechanism design model in which both the content and framing of the mechanism affect the agent's ability to manipulate the information he provides. The analysis is conducted in the context of a persuasion situation in which the listener announces the rules under which he will be persuaded by the speaker. The boundedly rational speaker is limited in his ability to find a persuasive story. He can do it only by using a certain inference method which depends on his real story and the pre-announced rules. Circumstances under which the listener's goal can be achieved and properties of the optimal persuasion rules are characterized.

The second author acknowledges financial support from ERC grant 269143.
We thank Noga Alon, Ayala Arad, Michael Richter, Rani Spiegler and especially Chuck Wilson for their comments.

1. Introduction

"I went to a bar and was told it was full. I asked the bar hostess what time one should arrive in order to get in. I was told by 12 and once the bar is full you can only get in if you are meeting a friend who is already inside. I lied that my friend was already inside. Without being told I would not have known which of the possible lies to tell in order to get in." (M.R. describing an actual experience at a Tel Aviv bar).

In this case, M.R. was trying to persuade the bar hostess to let him into the bar. The hostess revealed the circumstances under which she would be persuaded. In the absence of any means of verification, her statement informed M.R. how to persuade her.

Consider another example: You are interested in being a contestant in a TV game show with major prizes. To be eligible, you must not violate any of the following five criteria:

R1: If you have a sister and don't wear glasses,
then you must not be the oldest child in your family.

R2: If you don't wear glasses and you are the oldest child in your family,
then you must have a sister.

R3: If you don't have a sister,
then you must be the oldest child in your family.

R4: If you wear glasses and you are the oldest child in your family,
then you must not have a sister.

R5: If you have a sister and you are not the oldest child in your family,
then you must wear glasses.

Thus, someone who has a sister, does not wear glasses and is not the oldest child in his family violates R5. However, R5 guides him how to change his profile in order to get on the show. By simply changing his profile to "wears glasses" he can satisfy all the criteria. On the other hand, consider someone who does not have a sister, does not wear glasses and is not the oldest child. Only the antecedent of R3 is satisfied in his case. If he modifies his profile to satisfy the consequent of R3 and states that he is the oldest child in his family, he will then violate R2. It seems that the codex is a better guide to "altering the truth" and getting on the

show for someone with the first profile than for someone with the second.

Both of the above examples are persuasion situations. A persuasion situation involves a speaker and a listener. The speaker attempts to persuade the listener to take a certain action or to adopt a certain position. The interests of the two parties are not necessarily identical and depend on the speaker's "case" i.e. a body of relevant facts that only the speaker knows to be true or false. The speaker would like the listener to choose the desired action regardless of the true case, whereas the listener wishes to be persuaded only if certain conditions are met. In his attempt to persuade the listener, the speaker presents a "case", though not necessarily the true one. The listener is aware that the speaker may be providing false information but he cannot verify this one way or another.

A persuasion situation can be modeled as a leader-follower (listener-speaker) problem. First, the listener publicly announces and commits to a persuasion rule, i.e. a full description of the cases that, if presented to him by the speaker, will persuade him. Then, the speaker chooses a case to present. In order to persuade the listener the speaker can decide to present a false case. However, cheating effectively (i.e., in a way that induces the principal to behave according to the interests of the agent) can be difficult, as it requires the agent to invent a fictitious profile. Finding a "perfect lie" may require complex calculations, similar to those required in solving a puzzle or a system of equations.

Formally, let S be a set of feasible cases. The listener can choose between two actions: either accept the speaker's request or reject it. A subset of S, denoted by A, is the set of cases in which the listener, had he known the true case, would like to have granted the speaker's request. The residual set, $R = S - A$, consists of all the cases in which the listener would like to reject the speaker's request. The speaker, on the other hand, would like the listener to accept his request regardless of whether it is truth. The speaker knows the true case whereas the listener can only rely on the the speaker's statement in order to make his decision. One way of modeling the speaker's difficulties in cheating is to introduce a function M, where $M(s) \subseteq S$ is the set of cases that the speaker can present when the true case is s. (We chose this route in Glazer and Rubinstein (2006), where $M(s)$ was interpreted as the hard evidences that the speaker can present when the true case is s.) A persuasion rule is a set $P \subseteq S$, interpreted as

the set of cases in which the listener commits to choosing the speaker's requested action. Once a probability measure over the set *S* has been added, we can set the listener's objective to be the design of a persuasion rule that maximizes the probability that he will take the correct action (from his point of view), assuming that the speaker maximizes the probability that his request will be granted.

In the current paper, not only do we depart from the assumption commonly made in the mechanism design literature that cheating is easy, we also recognize that the speaker's ability to cheat effectively depends on the way in which the mechanism is constructed and framed. In particular, the speaker may sometimes use the persuasion rule itself as a guide in manipulating the information he provides to the listener. Therefore, the persuasion rule should be complex enough that a speaker whose case is in *R* will persuade the listener by manipulating the information but at the same time, simple enough that a speaker whose case is in *A* will be able to persuade the listener.

We model a persuasion rule as a set of conditions formulated in a certain language, referred to as a codex. A case is persuasive if it meets all the conditions in the codex. The ability of the speaker to present a false case depends not only on the true case and the set of cases that satisfy the codex, but also on the framing of the codex. We will characterize conditions under which the listener's acceptance set is truthfully implementable, in the sense that the outcome of the interaction is such that the listener accepts the speaker's request only when it should be accepted, and the speaker never lies. We also find conditions under which the listener's acceptance set is implementable, though not necessarily truthfully, in the sense that the outcome of the interaction is such that the request is accepted only when it should be accepted although the speaker might need to lie sometimes in order to persuade the listener.

2. The Model

The set of cases

Let $V = \{v_1, .., v_K\}$ be a set of $K \geq 2$ propositional variables. Each variable can take one of two truth values: "*T*rue" or "*F*alse". A *case* is a truth assignment for each of the variables. We assume that all 2^K cases are logically possible, namely that the content of the variables is such that the truth combination of some of the variables does not exclude the truth combination of

any of the others (as would be the case, for example, if the content of v_2 was the negation of the content of v_1). Denote by $s(v)$ the truth value of the variable v in the case s. In some cases, we will present a case s as a K-vector $(s_1,..,s_K)$ of 0's and 1's, where $s_k = 1$ ($s_k = 0$) means that the variable v_k takes the truth value T (F) in the case s. Let S be the set of all cases.

The speaker and the listener

There are two agents: a speaker and a listener. The speaker knows which case is true whereas the listener knows only the set S. The speaker wishes to persuade the listener to accept his request regardless of the true case. The listener can either accept or reject the request. He would like to accept the speaker's request only if the case belongs to a given set A. Let $R = S - A$ be the set of cases at which the listener would like to reject the speaker's request.

We analyze the following leader-follower scenario: First, the listener publicly commits to a codex, a set of conditions that the case presented by the speaker must satisfy in order to be accepted. Then, the speaker (who knows the true case) announces a case that may or may not be the true one. In making his decision, the listener is committed to apply the codex to the case announced by the speaker.

The codex

A *codex* is defined as a set of propositions in propositional logic that uses only the variables in the set V. We refer to a proposition in the codex as a *rule*. Only a case that satisfies all the propositions will "persuade" the listener. We make two assumptions regarding a codex:

1) Structure: Each rule φ in the codex must have the structure $\wedge_{y\in W}\varphi_y \rightarrow \varphi_x$ where W is a non-empty subset of V, $x \in V - W$ and each φ_v is either v or $-v$ (the negation of v).

2) Coherence: The codex does not contain conflicting rules, i.e., there is no pair of rules in the codex that lead to opposite conclusions. Formally, a codex is coherent if it does not contain two rules $\varphi = \wedge_{y\in W_1}\varphi_y \rightarrow x$ and $\psi = \wedge_{y\in W_2}\psi_y \rightarrow -x$ where for any $y \in W_1 \cap W_2$ we have $\varphi_y = \psi_y$. In other words, we require that there be no case under which two rules, in the codex, that lead to conflicting outcomes could be applied. To clarify, coherence does not only require that the codex will not contain the two rules $v_1 \rightarrow v_2$ and $v_1 \rightarrow -v_2$ but also that the codex will not contain the two rules $v_1 \rightarrow v_3$ and $v_2 \rightarrow -v_3$.

Going back to the TV show example described in the Introduction, and letting the three variables be S ="having a sister" G = "wearing glasses" and O="being the oldest child", then the codex presented in the introduction consists of the following five rules:

$S \wedge -G \rightarrow -O$, $-G \wedge O \rightarrow S$, $-S \rightarrow O$, $G \wedge O \rightarrow -S$ and $S \wedge -O \rightarrow G$.

Notation

For a proposition ψ, we use the notation $s \models \psi$ to represent the statement "proposition ψ is true in the case s" (for example, $s \models \wedge_{y \in I} \psi_y \rightarrow \psi_x$ if (i) there exists a variable $v \in I$ such that $\psi_v = v$ and $s(v) = F$ or $\psi_v = -v$ and $s(v) = T$ or (ii) for all $y \in I \cup \{x\}$ we have $s(y) = T$ iff $\psi_y = y$). Let $T(\psi)$ be the set of cases for which ψ is true, i.e., $T(\psi) = \{s|\ s \models \psi\}$.

For a codex Λ, let $T(\Lambda)$ be the set of cases that satisfy all propositions in Λ, i.e., $T(\Lambda) = \{s|\ s \models \varphi \text{ for all } \varphi \in \Lambda\} = \cap_{\varphi \in \Lambda} T(\varphi)$.

For any given rule $\varphi = \wedge_{y \in I} \varphi_v \rightarrow \varphi_x$, we denote $a(\varphi) = \wedge_{y \in I} \varphi_y$ (the antecedent of φ) and $z(\varphi) = \varphi_x$ (the consequent of φ).

Inference

The speaker can either state the true case or make up a false one. A fully rational speaker can come up with a case that satisfies the codex independently of what the true case is. We assume that the speaker is boundedly rational in the sense that he is limited in his ability to come up with a persuasive false case. The speaker cannot conceive of all cases but rather only the true case and the cases he can "infer" from it by using the codex. We say that, given Λ, **the speaker can infer** s' **from** s (denoted as $s \rightarrow_\Lambda s'$) if for every variable v for which $s'(v) \neq s(v)$, there is a rule $\varphi \in \Lambda$ such that:

(1) $s \models a(\varphi)$ and $s' \models a(\varphi)$

(2) $z(\varphi)$ is either v or $-v$ and

(3) $s' \models \varphi$ (that is, $s' \models z(\varphi)$).

In other words, given a codex Λ the speaker can infer s' from the true case s by keeping the truth values of some of the variables fixed and figuring out the values of the others by using those rules from the codex that refer in their antecedents only to the fixed variables. Of course, $s \rightarrow_\Lambda s'$ does not imply that s' satisfies the codex.

For example, suppose that $K = 4$ and Λ contains the two rules $v_1 \to -v_3$ and $v_2 \to -v_4$. Let $s = (1,1,1,1)$. The following four inferences are possible:

$s \to_\Lambda (1,1,1,1)$, $s \to_\Lambda (1,1,0,1)$, $s \to_\Lambda (1,1,1,0)$ and $s \to_\Lambda (1,1,0,0)$.

Lemma 1:

(a) The relation $\to_\Lambda$ is reflexive and anti-symmetric (i.e. for any distinct cases s and s', if $s \to_\Lambda s'$ then $s' \nrightarrow s$).

(b) If s is opposed to s' then $s \nrightarrow_\Lambda s'$ (s is opposed to s' if $s(v) \neq s'(v)$ for all v).

(c) If $s \to_\Lambda t$ and s' is between s and t then $s \to_\Lambda s'$ and $s' \to_\Lambda t$ (s' is between s and t if $s(v) \neq s'(v)$ implies that $s'(v) = t(v)$).

Given a reflexive and anti-symmetric binary relation $\to$, define $T(\to) = \{s \mid \text{for no } t \neq s,\ s \to t\}$.

Lemma 2: $T(\Lambda) = T(\to_\Lambda)$

Proof: Assume $s \notin T(\Lambda)$. Then, there is a rule $\varphi = \wedge_{y \in I} \varphi_y \to \varphi_x$ in Λ such that $s \models \varphi$ is not true, i.e., there is a rule φ such that s satisfies the antecedent $\wedge_{y \in I} \varphi_y$ but not the consequent φ_x. Thus, $s \to_\Lambda s'$ where s' is the case that differs from s only in the truth value of the variable x, i.e., $s \notin T(\to_\Lambda)$.

In the other direction, assume that $s \notin T(\to_\Lambda)$. Then, there is a different case t such that $s \to_\Lambda t$. In other words, there is a variable x and a rule $\varphi = \wedge_{y \in I} \varphi_y \to \varphi_x$ such that s and t satisfy φ's antecedent, $t(x) \neq s(x)$, and $t \models \varphi$. However, in that case s does not satisfy φ and therefore $s \notin T(\Lambda)$.

Persuasion

Our main assumption is that given the codex Λ, the speaker in s can persuade the listener if $s \to_\Lambda s'$ for some $s' \in T(\Lambda)$. Given a binary relation $\to$ let $P(\to) = \{s \mid \text{there is } t \in T(\to) \text{ such that } s \to t\}$. If $\to$ is reflexive, then $T(\to) \subseteq P(\to)$.

Define $P(\Lambda) = P(\to_\Lambda)$ and interpret $P(\Lambda)$ as the set of cases in which the speaker can persuade the listener. Note that it is possible that the speaker can infer more than one case from the true one (some persuasive and some not). By our definition, it is sufficient that the speaker can infer one persuasive case in order to be able to persuade the listener.

Implementation

The set A is implementable if there is a codex Λ such that $A = P(\Lambda)$.

The set A is truthfully implementable if there is a codex Λ such that $P(\Lambda) = T(\Lambda) = A$.

Thus, if a codex implements A then the speaker is able to persuade the listener in all cases in which the listener should be persuaded, but in none of the cases in which he should not. However, in some of the cases in which the listener should be persuaded the speaker has to "bend the truth" in order to persuade the listener. If a codex truthfully implements A, then the speaker whose case should persuade the listener is able to do so by simply telling the truth.

Comments:

(i) *An alternative definition of inference:* Let $V = \{v_1, v_2, v_3\}$ and $\Lambda = \{v_1 \to v_2,\ v_1 \to v_3\}$. Both $(1,0,0) \to_\Lambda (1,1,1)$ and $(1,0,0) \to_\Lambda (1,0,1)$. By our definition, the speaker can persuade the listener in the case $s = (1,0,0)$ since he can find a persuasive case $s' = (1,1,1)$ for which $s \to_\Lambda s'$ although he can also infer from s the case $(1,0,1)$, which is not persuasive. One could think of alternative definitions of inference, such as one where the speaker can persuade the listener only if *all* the cases that he can infer are within $T(\Lambda)$.

(ii) *The "revelation principle"*: The "revelation principle" is, of course, not valid in our framework. As we will see later, there are sets that are implementable but not *truthfully implementable*.

The neighborhood relation

A key element in the analysis is the neighborhood binary relation N on the set S. Define sNs' if s and s' differ in the truth value of exactly one variable. The relation N is symmetric and irreflexive. Two neighbors of the same case are not neighbors. We will refer to $d(s,s') = |\{v \mid s(v) \neq s'(v)\}|$ as the distance between s and s'.

A *path* is a sequence of distinct cases $(s_1,\ldots,s_L)$ such that $s_1Ns_2N\ldots Ns_L$. If $L > 2$ and s_LNs_1, then the path is a cycle. Note that any cycle contains an even number of cases and if $s_1Ns_2N_3$ then s_1 is not a neighbor of s_3. We say that a cycle is a *counting cycle* (referred to in

graph theory as a Hamiltonian Cycle) of the set X if it contains all elements of X. Obviously, S has a counting cycle $(s^1,\ldots,s^{2^K})$. A sequence $(s^0,s^1,,,\ldots,s^L)$ is a *ray* from s^0 if $s^{l+1}Ns^l$ and $d(s^l,s^0) = l$.

Let $N(s)$ be the set of neighbors of s. For any two cases s and s', $|N(s) \cap N(s')|$ is either 0 or 2. In particular, if s' and s'' are two neighbors of s, then there is exactly one more case $t \neq s$ such that tNs' and tNs''.

Complete rules

A *complete rule* is one of the type $\wedge_{y\in V-\{x\}}\ \varphi_y \rightarrow \varphi_x$. In other words, its antecedent refers to $K-1$ variables and the consequent to the remaining one. If a codex Λ includes the complete rule $\wedge_{y\in V-\{x\}}\ \varphi_y \rightarrow \varphi_x$ then $s \rightarrow_\Lambda s'$ where s and s' are the two neighbors defined by $s \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge -\varphi_x$ and $s' \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge \varphi_x$. For any two neighbors s and s', let $R(s,s')$ be the complete rule $\wedge_{y\in V-\{x\}}\ \varphi_y \rightarrow \varphi_x$ where both $s \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge -\varphi_x$ and $s' \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge \varphi_x$.

Canonical codexes

A *canonical codex* is one that consists only of complete rules. Each rule excludes exactly one distinct case. If a canonical codex implements the set A, then the number of its rules is at least equal to the number of cases in R and thus is typically large. Thus, a canonical codex often lacks a natural interpretation though it is analytically useful. Note also that a canonical codex makes the inference operation for a case in A relatively simple since the most the speaker has to do is check the K neighboring cases.

Note that the codex structure assumed here allows the specification of any subset $X \subseteq S$:

Lemma 3: For every set $X \subseteq S$, there is a (canonical) codex Λ such that $T(\Lambda) = X$.

Proof: Let $(s^1,,,\ldots,s^L)$ be a counting cycle of S. The codex $\Lambda = \{R(s^l,s^{l+1}) \mid s^l \notin X\}$ is canonical and coherent and clearly, $T(\Lambda) = X$.

3. Examples

Example 1: The set S is truthfully implementable by the empty codex. The empty set is implementable by the codex Λ, which contains all rules $R(s^l, s^{l+1})$ where $(s^1,,,\ldots,s^{2^K})$ is a *counting cycle* of S. Obviously, $T(\Lambda) = \emptyset$ and thus $P(\Lambda) = \emptyset$ as well. This implementation is truthful in a degenerate sense.

Example 2: This example demonstrates the critical role of the framing of the codex. Let $K = 3$ and $A = \{(1,1,1),(0,0,0)\}$. The following table presents three codexes, Λ_1, Λ_2, and Λ_3, which induce the same set of persuasive cases (namely, $T(\Lambda_i) = A$ for all i). However, the codexes differ in $P(\Lambda_i)$, the sets of cases in which the speaker can persuade the listener. Only Λ_3 (truthfully) implements A.

Λ	v_2 and v_3 as v_1	v_2 as v_1; v_3 as v_2	$2T \rightarrow 1T$ and $1T \rightarrow 2T$
	$v_1 \rightarrow v_2$	$v_1 \rightarrow v_2$	$-v_i \wedge v_{i+1} \rightarrow v_{i+2}$ $(\forall i)$
	$v_1 \rightarrow v_3$	$-v_1 \rightarrow -v_2$	$v_i \wedge -v_{i+1} \rightarrow -v_{i+2}$ $(\forall i)$
	$-v_1 \rightarrow -v_2$	$v_2 \rightarrow v_3$	
	$-v_1 \rightarrow -v_3$	$-v_2 \rightarrow -v_3$	
$T(\Lambda)$	A	A	A
$P(\Lambda)$	S	$S - \{\{1,0,0),(0,1,1)\}$	A

Example 3: Let $K = 3$ and $A = \{(1,0,0),\ (0,1,0),\ (0,0,1)\}$. We will see that A is not implementable. Assume that Λ implements A.

Case (1): $T(\Lambda) = A$. The case $(0,0,0)$ is not in $T(\Lambda)$ and hence there is a rule in Λ that this case does not satisfy and w.l.o.g. the rule is either $-v_1 \rightarrow v_3$ or $-v_1 \wedge -v_2 \rightarrow v_3$. If $-v_1 \rightarrow v_3$ is in the codex then $(0,1,0) \notin T(\Lambda)$. If $-v_1 \wedge -v_2 \rightarrow v_3$ is in the codex, then $(0,0,0) \rightarrow_\Lambda (0,0,1)$ and hence $(0,0,0) \in P(\Lambda)$ although $(0,0,0) \notin A$. In either case, we arrive at a contradiction.

Case (2): One of the cases in A, w.l.o.g. $(0,0,1)$, is not in $T(\Lambda)$. Then, there must be another case in A, w.l.o.g. $(0,1,0)$, such that $(0,0,1) \rightarrow_\Lambda (0,1,0)$. This requires that $-v_1 \rightarrow v_2$ be in the codex. However, in that case, $(0,0,0) \rightarrow_\Lambda (0,1,0) \in T(\Lambda)$ and therefore $(0,0,0) \in P(\Lambda)$, a

contradiction.

Example 4: Let A consist of all cases except for the K cases in which exactly one variable receives the value T. The set A is implemented (although its complement is not implementable) by the codex Λ that consists of $K(K-2)$ rules: $v_i \rightarrow v_j$ where $j \neq i+1$ ($K+1$ is taken as 1). Obviously, $T(\Lambda) = \{all\ F, all\ T\}$. The codex allows the speaker to infer the "*all T*" case from every case in A except for "*all F*". For any case in R, where only one variable v_i receives the value T, the speaker can only infer cases in which v_{i+1} is F, and they do not satisfy the codex.

Example 5: Let $A_m = \{s|\ s$ receives the value T for at least m variables$\}$ where $0 < m < K$. We will show that A_m is implementable.

Let Λ be the codex containing all rules of the type $[\wedge_{v \in W} v] \wedge [\wedge_{v \in X-W-\{y\}} -v] \rightarrow y$ where W is a set of at most m variables and $y \notin W$. The codex requires that if from among $K-1$ variables at most m variables receive the value T, then the $K'th$ variable should receive the value T as well. $T(\Lambda) = A_{m+1}$ and $P(\Lambda) = A_m$. Thus, the speaker whose case assigns the truth value T to exactly m variables has an incentive to slightly exaggerate and claim that there are $m+1$ true variables. This implementation is not truthful, but as will be shown later in Proposition 3, A_m is truthfully implementable for $K > 3$ and $m > 2$.

Example 6: For $K = 2$ all sets are implementable except for the four singletons and the two sets each consisting of two opposing cases. To see this it is sufficient w.l.o.g. to consider the following sets:

(a) $A = \{(1,0),(1,1))\}$ is implementable by $\Lambda = \{-v_1 \rightarrow -v_2,\ \ -v_2 \rightarrow v_1,\ \ v_1 \rightarrow v_2\}$, a codex which induces the inference relation: $(0,1) \rightarrow_\Lambda (0,0) \rightarrow_\Lambda (1,0) \rightarrow_\Lambda (1,1)$. Clearly, $T(\Lambda) = \{(1,1)\}$ and $P(\Lambda) = A$.

(b) $A = S - \{(0,0)\})$ is implementable by the codex $\Lambda = \{-v_1 \rightarrow v_2,\ \ v_2 \rightarrow v_1\}$ which induces the inference relation $(0,0) \rightarrow_\Lambda (0,1) \rightarrow_\Lambda (1,1)$. Thus, $T(\Lambda) = \{(1,1),(1,0)\}$ and $P(\Lambda) = A$.

(c) $A = \{(1,1)\}$ is not implementable. Assume that Λ implements A. It must be that $P(\Lambda) = T(\Lambda) = \{(1,1)\}$. The codex excludes the case $(0,0)$ and thus (w.l.o.g.) $-v_1 \rightarrow v_2$ is in the codex. The case $(0,1)$ has to be excluded. By the consistency requirement $-v_1 \rightarrow -v_2$ is

not in the codex. Thus, it must be that $v_2 \to v_1$ is a rule in the codex. However, in that case, $(0,1) \to_\Lambda (1,1)$ and $(0,1) \in P(\Lambda)$, thus contradicting $P(\Lambda) = A$.

(d) $A = \{(1,0),(0,1))\}$ is not implementable. Assume that Λ implements A. Since A is composed of two opposing cases one cannot be inferred from the other and thus it must be that $P(\Lambda) = T(\Lambda) = A$. The case $(1,1)$ is excluded, i.e., w.l.o.g. $v_1 \to -v_2$ is a rule in the codex. However, in that case, $(1,1) \to_\Lambda (1,0)$ and thus $(1,1) \in P(\Lambda)$, a contradiction.

3. Truthful Implementability

In this section we fully characterize the truthfully implementable sets.

Proposition 1: If the set A is truthfully implementable, then it is truthfully implementable by a canonical codex.

Proof: Let Λ be a codex such that $T(\Lambda) = P(\Lambda) = A$.

By Lemma 2, $T(\Lambda) = T(\to_\Lambda)$ and thus for every $s \in R$ there is a case $t \neq s$ such that $s \to_\Lambda t$. Let $n(s)$ be a neighbor of s which is between s and t. By Lemma 1, we have $s \to_\Lambda n(s) \to_\Lambda t$ and therefore $n(s) \notin T(\Lambda)$. The canonical codex $\Lambda' = \{R(s,n(s)) \ \ | s \in R\}$ truthfully implements A.

We say that a set of cases C is *connected* if for any two cases $s, s' \in C$ there is a path of elements in C connecting s and s'. C is a connected component of R if it is a maximal connected subset of R.

Proposition 2: The set A is truthfully implementable if and only if every connected component of R contains a cycle.

Proof: Assume that A is truthfully implementable. By Proposition 1, the set is implementable by a canonical codex Λ. Then, for every $s \in R$ there is a case $s' \in R$ such that sNs' and $s \to_\Lambda s'$. Let s_1 be an arbitrary element in R. Since Λ implements A there is $s_2 \in R$ such that $s_1 \to_\Lambda s_2$. Continuing in this manner, we obtain $s = s_1 \to_\Lambda s_2 \to_\Lambda \ldots \to_\Lambda s_L = s'$ where $s_L = s_{L'}$ for some $L' < L$ and $s_l \in R$ for all l. Thus, also $s = s_1 N s_2 N .., N s_L = s'$ with $s_{L'} = s_L$. In other words, s_1 is connected in R to a cycle of elements in R.

In the other direction, assume that any connected component of R has a cycle. Define the

binary relation $\rightarrow$ on R as follows: Let C be a connected component of R. Select a subset of cases in C that form a cycle $s_1 N s_2 N .., N s_L N s_1$. For any l, add $s_l \rightarrow s_{l+1}$ to the relation (identify $L+1=1$). For any element $s \in C - \{s_1 \ldots, s_L\}$, choose a shortest path $t_1 N t_2 .., N t_N$ where $t_1 = s$ and and t_N is in the cycle and add $t_1 \rightarrow t_2$ to the relation. Obviously, the relation $\rightarrow$ is anti-symmetric. Let $\Lambda = \{R(s,s') \mid s \rightarrow s'\}$. The relation $\rightarrow_\Lambda$ is identical to $\rightarrow$ and $P(\Lambda) = T(\Lambda) = A$.

The following proposition describes families of acceptance sets that are truthfully implementable. The first family consists of all sets that are "small" in the sense that they contain no more than $K-1$ cases. Each of the sets in the second family consists of all cases for which the number of variables that are true exceeds a certain threshold. The sets belonging to the third family have the property that a particular variable is true (or false) in all cases included in the set. The last family consists of all sets for which there are two variables, such that the inclusion of a case in the set is independent of their truth values. These two "degenerate" variables are used in the codex merely in order "to confuse" the undeserving speaker.

Proposition 3: For $K \geq 3$, any set A that satisfies at least one of the following conditions is truthfully implementable:

(1) A is "small": it consists of at most $K-1$ cases.

(2) The number of true variables must exceed a threshold: There exists a number $m \geq 3$, such that $A = A_m = \{s|$ at least m variables receive the value T at $s\}$.

(3) There is a variable that must be true (or false): There exists a variable v such that $A \subseteq T(v)$ (or $T(-v)$). (Recall that $T(v)$ is the set of all cases in which v receives the value T).

(4) There are two irrelevant variables v' and v'' such that if $s \in A$, then so is any case s' for which $s(v) = s'(v)$ for all v other than v' and v''.

Proof: Due to Proposition 2, it is sufficient to show that every $s \in R$ has a path in R connected to a cycle in R.

(1) First, we show that the set R is connected. This is done by verifying that for any two cases s and t in R that are not neighbors, there are K "disjoint" paths connecting s and t. Since A contains at most $K-1$ elements, at least one of the paths contains only elements of R. Thus,

R is connected.

Second, we show that R contains a cycle. Otherwise, let $s_1 N s_2 N \ldots N s_L$ be a longest path of distinct elements in R. Since R contains more than half of the cases, there must be two opposing elements belonging to R and thus $L \geq K + 1 \geq 4$.

Since $s_3 \in N(s_2) \cap N(s_4)$ there is another case r such that $s_2 N r N s_4$. The case r must be in A since otherwise (s_2, s_3, s_4, r) forms a cycle in R. The case r is not a neighbor of s_1 since s_1 is a neighbor of s_2. The set $N(s_1)$ consists of $s_2 \in R$ and $K - 1$ other cases. It is impossible that all of them are in A since r is not one of them. Thus, $N(s_1)$ contains another element in R (in addition to s_2) and we can extend the path.

(2) R is connected since each case in R is connected to the "all F" case. The set R contains the $2K$-element cycle

$((1,0,\ldots,0),(1,1,0,\ldots,0),(0,1,0,\ldots,0),(0,1,1,0,\ldots,0)\ldots,(0,0,\ldots,1),(1,0,\ldots,0,1))$

(3) The set $T(-v) \subseteq R$ is obviously connected and contains a cycle. Any element in R is either in $T(-v)$ or is a neighbor of a case in $T(-v)$. Thus, R is connected and contains a cycle.

(4) Any $s \in R$ is part of a cycle containing the four-case set $\{t|\ t(v) = s(v)$ for any $v \notin \{v', v''\})$.

An alternative interpretation of truthful implementation: Let $K = 3$ and let $\Lambda = \{v_1 \rightarrow v_2,\ v_2 \rightarrow v_3\}$. Then $(1,0,0) \rightarrow_\Lambda (1,1,0)$ and $(1,1,0) \rightarrow_\Lambda (1,1,1)$. However, by our assumptions, the speaker cannot make a chain of inferences and thus cannot infer the persuasive case $(1,1,1)$ from $(1,0,0)$. Had we allowed the speaker to make a chain of inferences, the following alternative definition of implementation would have applied:

There exists a codex Λ such that:

(i) for any $s \in A$ there is a chain $s = s_1 \rightarrow_\Lambda s_2 \ldots \rightarrow_\Lambda s_L$ where $s_L \in T(\Lambda)$.

(ii) for no $s \in R$ does there exists a chain $s = s_1 \rightarrow_\Lambda s_2 \ldots \rightarrow_\Lambda s_L$ where $s_L \in T(\Lambda)$.

Lemma 4: The set A is implementable in the alternative sense if and only if it is truthful implementable.

Proof: If A is truthful implementable then there exists a codex Λ such that $P(\Lambda) = T(\Lambda) = A$. Thus, part (i) of the alternative definition is satisfied since for any $s \in A$,

$s \to_\Lambda s \in T(\Lambda)$. Part (ii) is satisfied since there is no $s \in R$ such that $s \to t \in T(\Lambda)$. Thus, A is implementable in the alternatives sense.

On the other hand, assume that Λ implements the set A in the alternative sense. By (ii), no members of R is in $T(\Lambda)$. For any $s \in R$ there exists s' such that $s \to_\Lambda s'$ and w.l.o.g. $s'Ns$. The case s' is not in A since otherwise, by (i) there would be a chain $s' = s_1 \to_\Lambda s_2 \ldots \to_\Lambda s_L$ with $s_L \in T(\Lambda)$ and then $s \to_\Lambda s_1 \to_\Lambda s_2 \ldots \to_\Lambda s_L$, contradicting (ii). Consider the codex $\Lambda' = \{R(s, s') \mid s \in R\}$. Then, $P(\Lambda') = T(\Lambda') = A$.

4. Implementability (not necessarily truthful)

Proposition 4: A set A is implementable by a canonical codex if and only if there is a reflexive binary relation $\to$ satisfying:

(1) anti-symmety

(2) $P(\to) = A$ and

(3) for any $s \neq s'$, if $s \to s'$ then sNs'.

Proof: Assume that A is implementable by a canonical codex Λ. The relation $\to_\Lambda$ satisfies properties (1,2,3): the coherence of the codex implies (1); the implementability of A by the codex is equivalent to (2); and the codex being canonical implies (3).

On the other hand, given a relation $\to$ that satisfies (1,2,3), consider $\Lambda = \{R(s, s') \mid s \neq s'$ and $s \to s'\}$. (1) implies that the codex is coherent and (3) implies that the codex is canonical. The relation $\to_\Lambda$ is equivalent to $\to$ and by (2) we have $P(\Lambda) = P(\to_\Lambda) = P(\to) = A$.

The next result is analogous to Proposition 1. A necessary and sufficient condition for implementability is implementability by a canonical codex.

Proposition 5: If the set A is implementable, then it is implementable by a canonical codex.

Proof: Let Λ be a codex that implements A. We seek a reflexive binary relation satisfying (1,2,3):

The relation $\to_\Lambda$ is reflexive and satisfies (1,2) and in addition has the following property:

(4) Betweenness: If $s \to s'$ and t is a case "between" s and s', then $s \to t \to s'$.

First, define a new reflexive relation $\to$ such that:

(a) For $s \in A - T(\Lambda)$, choose one case $s' \in T(\Lambda)$ such that $s \to_\Lambda s'$ and define $s \to s'$.

(b) For $s \in R$, choose a case $s' \neq s$ for which $s \to_\Lambda s'$. Since $\to_\Lambda$ satisfies (4) there is $s' \in N(s)$ for which $s \to_\Lambda s'$. Since $s \notin P(\Lambda)$, $s' \notin T(\Lambda)$. Define $s \to s'$.

Since $\to_\Lambda$ satisfies (1,2) so is the relation $\to$ which satisfies also the following property:

(5) For any $s \in R$ the outgoing case is a neighbor and for any $s \in A$, if $s \to s'$, then all elements between s and s' are in A.

We now modify the relation $\to$ recursively as follows:

(i) if $s \in A - T(\to)$ and $N(s) \cap T(\to) \neq \emptyset$ choose a case s' in that set and define $s \to s'$.

(ii) Let $s \to s'$ where $s \in A$ and $s' \notin N(s)$. Then s has a neighbor s'' between s and s' and $s'' \in A$ and therefore there exists $s''' \in T(\to)$ such that $s'' \to s'''$. Delete $s'' \to s'''$ and $s \to s'$ from the relation and add $s \to s''$. If there is a case $r \to s''$ where $r \in R$, then s'' and r are neighbors. Since both s and r are neighbors of s'' there is another case t which is a joint neighbor to s and r. By (i), $t \notin T(\to)$. If $t \in A$, then add $r \to t$. If $t \in R$, then delete $t \to t'$ (t' could be r!!) and add $r \to t$ and $t \to s$. The new relation satisfies (1), (2) and (5) but with one less element in A that goes to a non-neighbor.

Go back to (i). Following a finite number of iterations we obtain a relation satisfying (1,2,3).

Propositions 6 and 7 present families of sets that are and are not implementable, respectively. Note that the propositions do not fully characterize the implementable sets.

Proposition 6: For $K \geq 3$, any set A that satisfies at least one of the following conditions is implementable:

(1) $A \supseteq T(v)$ for some variable v (recall that $T(v)$ is the set of all cases in which the variable v receives the value T).

(2) $A \supseteq B$ where B is a truthful implementable set and every case in $A - B$ is a neighbor of a case in B.

(3) $|R| \leq K$.

(4) A is monotonic in the following sense: if $s \in A$ and s' is a case such that whenever v is true in s it is also true in s', then $s' \in A$.

(5) A's connected components are all "radius 1" sets (B is "a radius 1" set if it is not a singleton and there is a case $b \in B$ such that all other elements in B are neighbors of b).

(6) Every case has a neighbor in A and every case in R has also a neighbor in R.

Proof: For each of the five cases, we will construct a relation $\rightarrow$ and it will be straightforward to verify that it satisfies properties (1,2,3) in Proposition 4:

(**1**) Let $s_1,..,s_{2^{K-1}}$ be a counting cycle of $T(-v)$.

The relation $\rightarrow$ contains for any $s_l \in T(-v)$ one pair "$s_l \rightarrow x$":

If $s_l \in R$ then $x = s_{l+1}$ and

if $s_l \in A$ then x is s_l's neighbor in $T(v)$.

Note that $T(\rightarrow) = T(v)$ and $P(\rightarrow) = A$.

(**2**) The proof is very similar to the proof of part (1) above.

(**3**) Distinguish between three cases:

(a) There is a case $r^* \in R$ such that $d(r,r^*) \leq 1$ for all $r \in R$.

For every $r \in R$, let $(r^*, r, n(r), n^2(r))$ be a ray. Both $n(r)$ and $n^2(r)$ must be in A. Construct the relation $\rightarrow$ such that for any $r \in R$: $r^* \rightarrow r$, $r \rightarrow n(r)$ and $n(r) \rightarrow n^2(r)$. Obviously, $T(\rightarrow) = A - \{n(r) \mid r \in R\}$.

(b) There is no $s^* \in A$ such that $d(r,s^*) \leq 2$ for all $r \in R$.

For every $r \in R$, let $n(r)$ be one of r's neighbors in A (which exists since $|R| \leq K$). Let $M = \{n(r)\}_{r \in R}$. The case $n(r)$ has K neighbors, one of which, $n^2(r)$, must not be a member of $M \cup R$ (otherwise $d(n(r),s) \leq 2$ for all $s \in R$). Construct the relation $\rightarrow$ such that $r \rightarrow n(r)$ and $n(r) \rightarrow n^2(r)$ for every $r \in R$. Note that $T(\rightarrow) = A - M$ and the relation satisfies (1,2,3).

(c) There is a case $s^* \in A$ such that $d(r,s^*) \leq 2$ for all $r \in R$ and there is a case $r^* \in R$ such that $d(r,r^*) \leq 1$ for all $r \in R$.

For every $r \in R$ such that $d(s^*,r) = 2$, choose a path $(s^*, n(r), r)$. If $n(r) \in R$, add $r \rightarrow n(r)$ to the relation. If $n(r) \in A$, add $r \rightarrow n(r)$ and $n(r) \rightarrow s^*$ to the relation.

For every $r \in R$ such that rNs^*, there is a case $n(r) \in A$ such that $d(s^*, n(r)) = 2$ ($n(r)$ has $K-1$ neighbors at a distance of 2 from s^* and since this is not case (a) one of them must not be

in R). Let $n^2(r)$ be a neighbor of $n(r)$ such that $d(s^*, n^2(r)) = 3$. Add $r \to n(r)$ and $n(r) \to n^2(r)$.

Note that $T(\to) = A - \{n(r)\}_{r \in R}$.

(4) In example 1, we dealt with the cases $A = \emptyset$ and $A = S$. The case $A = \{alltruth\}$ is dealt with in Proposition 3(1). For any other set A that satisfies montonicity, $allfalse \in R$ and $alltruth \in A$ and so is at least one of its neighbors, denoted by s'. Let $s^0, \ldots, s^K$ be a ray from $s^0 = allF$ to $s^K = alltruth$ with $s^{K-1} = s'$. Let k^* be the minimal k such that $s^k \in A$. Define a relation $\to$ such that $s^k \to s^{k+1}$ for all $0 \le k \le k^*$. For any $r \in R$ such that only one variable receives the value T and is not on the ray, add $r \to allfalse$ to the relation. For any other $r \in R$ that is not on the ray, let r^- be a neighbor of r not on the ray with one less variable receiving the value T. Such an r^- exists since r has at least two neighbors with one less true variable and at most one of them is on the ray. Add $r \to r^-$ to the relation. $T(\to) = A - \{s^{k^*}\}$.

(5) Construct the relation $\to$ as follows:

For each connected component of the set A, designate a case that is a neighbor of all other elements in the component as a "center". Denote by A^* the set of all such centers.

For each $s \in A - A^*$ add $s \to s^*$ to the relation.

For each $s \in R$ that has a neighbor s' in $A - A^*$, add $s \to s'$ to the relation.

For each $s \in R$ such that all its neighbors in A are centers, let c be one of those centers. The case c has a neighbor $c' \in A - A^*$. Thus, both s and c' are neighbors of c. Let x be the other element in $N(s) \cap N(c')$. The case x is in R since otherwise x would be connected to c in A and $d(x, c) = 2$. Add $s \to x$ and note that x has a neighbor in $A - A^*$.

Finally, for each $s \in R$ that does not have a neighbor in A, add $s = s^l \to s^{l+1}$ to the relation (where (s^l) is a fixed counting cycle of S).

We thus have $T(\to) = A^*$.

(6) We label a case s as even or odd according to the parity of the number of variables that receive the value T in s.

Let A^* be the set of all even cases in A. Construct the relation $\to$ as follows:

For each odd $s \in A$, there is an even neighbor $t \in A$ (and thus $t \in A^*$). Add $s \to t$ to the relation.

For every even $r \in R$, there is an odd neighbor $t \in A$. Add $r \to t$ to the relation.

For every odd $r \in R$, there is an even neighbor $t \in R$. Add $r \to t$ to the relation.

Clearly, $T(\to) = A^*$.

Proposition 7: The following are families of sets that are not implementable:

(1) There exists $s^* \in R$ such that $A \supseteq N(s^*)$ and for any $x \in N(s^*)$ we have $N(x) \subseteq R$.

(2) All connected components of A are singletons and A is not truthfully implementable.

Proof:

(1) W.l.o.g. $s^* = allF$. If there were a codex that implements A, then by Proposition 4 there would also be a canonical codex Λ that implements A.

It must be that $allF \to_\Lambda 1_v$ (a case that assigns T only to v). $1_v \in A - T(\Lambda)$. It must be that $1_v \to_\Lambda s$ where s is a neighbor of 1_v and $s \in T(\Lambda) \subseteq A$. However, by assumption, 1_v does not have neighbors in A, a contradiction.

(2) Assume that A is implementable and thus is also implementable by a canonical codex Λ, though not truthfully implementable. Then, there is a case $s \in A$ that is not in $T(\Lambda)$. It must be that there is a case $s' \in T(\Lambda)$ such that $s \to_\Lambda s'$ and $s'Ns$. Thus, the connected component in A that contains s also contains s'.

Corollary: For $K = 3$:

(a) A is not implementable if and only if it consists of 3 or 4 "isolated" cases.

(b) A is truthfully implementable if and only if either (i) A is a subset of $T(v)$ or $T(-v)$ for some variable v or (ii) A consists of two opposing cases.

Proof:

(a) By Proposition 3(1) any set A such that $|A| \leq 2$ is implementable.

Consider a set A that contains two opposing cases: w.l.o.g. *allfalse* and *alltruth*. The set $S - \{allflase, alltruth\}$ has a counting cycle $s^1, .., s^6$. The following 6-rule canonical codex truthfully implements A: For every s^l, if $s^l \in R$ add $R(s^l, s^{l+1})$ to the codex and if $s^l \in A$ then one of the elements $s^l_* \in \{allfalse, alltruth\}$ is a neighbor of s^l and then add $R(s^l, s^l_*)$ to the

codex.

Consider a set A that contains two neighboring cases, c^1 and c^2, and does not contain any pairs of opposing cases. Let $(s^1,\ldots,s^6)$ be the counting cycle of $S-\{c^1,c^2\}$. Each $s \in A \cap \{s^1,..,s^6\}$ is a neighbor of $\alpha(s) \in \{c^1, c^2\}$. A codex that implements A would consist of of six rules: $R(s^l,s^{l+1})$ for any $s^l \in R$ and $R(s^l,\alpha(s^l))$ for any $s^l \in A$.

We are left with the possibility that A contains 3 or 4 isolated cases. Then, there is a case $r \in R$ such that all of its neighbors are in A and then by Proposition 2 it is not truthfully implementable and by Proposition 7(2) is not implementable either.

(b) For A consisting of two opposing cases, the claim follows from the proof of part (a). For the case in which A is a subset of $T(v)$ or $T(-v)$, the claim follows from Proposition 3(3).

On the other hand, if A is truthfully implementable, then by Proposition 2 the set R contains a cycle that must be one of the following three types:

(i) A cycle containing 4 cases, isomorphic to $T(v_1) = \{(1,1,1),(1,1,0),(1,0,0),(1,0,1)\}$. Then A is a subset of some $T(-v_1)$.

(ii) A cycle containing 6 cases. Then A is a set of at most two cases and if it does not consist of two opposing cases, then the two cases have the same truth value for at least one variable.

(iii) The entire set S and then $A = \emptyset$.

5. Discussion

5.1. Preliminary Experimental Evidence

We obviously do not view the bounded rationality element in our model as an exact description of reality. Nevertheless, we believe that it captures some elements of real life. In order to test this intuition we conducted the following experiment: Subjects from more than 30 countries who had all taken a game theory course prior to 2008 were asked to participate in a short web-based experiment. The subjects were first asked three "Yes or No" questions: "Do you have a sister"", "Do you wear glasses?" and "Are you the oldest child in your family?" After a short training they were presented with the codex described in the Introduction. They

were then asked whether they were eligible to be on the show. The 59 subjects (out of 232) who answered "yes" ended their participation in the experiment at this point. 88% of these 59 subjects answered correctly, according to their declared profile (namely their true profile was indeed one of the two cases in $T(\Lambda)$, i.e. *FTT* or *TTF*. The subjects who answered that they were not eligible to be on the show were asked one additional question:

"You indicated that you are not eligible to be on the show, i.e. that you do not satisfy at least one of the five criteria [presented again on the screen]. How would you answer the following questions if you wanted to become eligible for the show [the students were then presented again with the three profile questions]?" A prize of $100 was promised to one of the participants whose answer make him eligible.

Only 12% out of 183 subjects who had declared that they were not eligible gave the wrong answer given their true profile. The other 160 subjects can be divided into three groups:

Inconsistent: There were 18 subjects who despite having answered that they are not eligible gave the same answer again. They were either confused or misunderstood the question.

$P(\Lambda) - T(\Delta)$: There were 64 subjects whose declared profile was one of the following three cases: *TTT*, *TFF* or *FTF*.

$S - P(\Lambda)$: The remaining 58 subjects had a declared profile that was one of the following three cases: *FFF*, *FFT* or *TFT*.

We conjectured that the success rate of the subjects in $P(\Lambda) - T(\Delta)$ would be higher than that of the subjects in $S - P(\Lambda)$. The results weakly supported our conjecture: 15.6% of the subjects in $P - T$ failed to cheat effectively while 24.1% of the subjects in $S - P(\Lambda)$ failed to do so. The more striking result relates to response time: the median response time (MRT) of the subjects in $P(\Lambda) - T(\Delta)$ who cheated effectively was 157 seconds vs. 218 seconds for subjects in $S - P(\Lambda)$ who cheated effectively. The difference of one minute indicates that the subjects in $S - P(\Lambda)$ found it more difficult to cheat effectively. Incidentally, the MRT of the inconsistent subjects was only 42 seconds.

5.2. *Related Literature*

The idea that cheating is difficult is, of course, not a new one. Within the economic literature, it appears, for example, in Kamien and Zemel (unpublished, 1990). They reinterpreted Cook's Theorem (see Cook (1971)), which states that deciding whether a given

Boolean formula in conjunctive normal form has an assignment that makes the formula true is NP complete.

Kartik (2009) studies a model of persuasion in which a speaker incurs a cost in order to misrepresent private information. He shows that inflated language naturally arises in this environment, such that the speaker claims to be of a higher type than he would have under complete information.

The idea that the framing of a mechanism may also provide some guidance to the participants appeared in (the completely ignored) Glazer and Rubinstein (1996). In that paper, we introduced the concept of "guided iterative elimination of dominated strategies" and showed that it is equivalent to implementation using a subgame perfect equilibrium of an extensive game with perfect information.

The idea that the mechanism itself can affect agents' preferences and thus influence the implementability of social outcomes appears in Glazer and Rubinstein (1998). In that paper, a number of experts receive noisy signals regarding a public decision. Two "cultures" were compared: In the first, the experts are driven only by the public motive to increase the probability that the desirable action will be taken. In the second, each expert is also driven by a private motive to have his recommendation adopted. We show that only the second culture gives rise to a mechanism whose unique equilibrium outcome achieves the public target.

A model of implementation with Bounded Rationality can be found in Eliaz (2002) who investigates the implementation problem when some of the players are "faulty" in the sense that they fail to act optimally. He introduces a solution called "fault tolerant implementation", which requires robustness to deviations from equilibrium. It shows that under symmetric information any choice rule that satisfies certain properties can be implemented if the number of faulty players is sufficiently small.

5.3. *Conclusion*

The model presented here facilitates the discussion of some basic considerations used by a principal attempting to elicit information from an agent who may have an incentive to cheat. The principal would like the mechanism to be complex enough that an agent, whose interests clash with his own, will not be able to infer from the mechanism itself how to successfully distort the information he conveys. At the same time, he would like the mechanism to be

simple enough that an agent whose interests coincide with his own will be able to collaborate with him.

Following are some of the main insights of the paper:

1) In some cases, it is optimal for the listener to use a codex that will help the speaker "alter the truth", that is, present a false but persuasive case. This result is consistent with the casual observation that some exaggeration is sometimes viewed as necessary in real life.

(2) If the circumstances under which the listener should (from his point of view) accept the speaker's request are rare, then truthful implementation is easy. This will be accomplished by means of a codex that will trap all "undeserving" speakers (i.e. speakers whose case should not be accepted) in a "circle of lies." In other words, an undeserving speaker is (mis)guided by the codex to pretend to be another undeserving speaker whose case is rejected by the codex and who, in turn, is guided by the codex to pretend to be a third undeserving speaker whose case is rejected and so on. This procedure continues until one of the undeserving speakers is guided by the codex to present a case that appears previously in the chain. This "trick" is used in all mechanisms that achieve truthful implementation.

(3) If the circumstances under which the listener should reject the speaker's request are rare, then the optimal mechanism may require some of the deserving speakers to (successfully) cheat. The codex, in this case, will actually guide an undeserving speaker to pretend to be a deserving one whose case is rejected by the codex while the deserving speaker is guided by the codex to pretend to have a case that is accepted by the codex. This kind of trick is used in all the cases where untruthful implementation is feasible but truthful implementation is not.

In general, our model is not intended to closely mirror real-life situations. Nevertheless, it should suggest a new direction for research into mechanism design with boundedly rational agents.

References

Cook, Stephen A. (1971). "The Complexity of Theorem Proving Procedures". *Proceedings Third Annual ACM Symposium on Theory of Computing*, 151-158.

Eliaz, Kfir (2002). "Fault Tolerant Implementation". *Review of Economic Studies*, 69, 589-610.

Glazer, Jacob and Ariel Rubinstein (1996). "An Extensive Game as a Guide for Solving a Normal Game". *Journal of Economic Theory*, 70, 32-42.

Glazer, Jacob and Ariel Rubinstein (1998). "Motives and Implementation: On the Design of Mechanisms to Elicit Opinions". *Journal of Economic Theory*, 79 (1998), 157-173.

Glazer, Jacob and Ariel Rubinstein (2006). "A Study in the Pragmatics of Persuasion: A Game Theoretical Approach". *Theoretical Economics*, 1, 395-410.

Kartik, Navin (2009). "Strategic Communication with Lying Costs". *Review of Economic Studies*, 76, 1359-1395.

Keimen , Morton I. and Eitan Zemel (1990). "Tangled Webs: A Note on the Complexity of Compound Lying". (mimeo)