AgEcon SEARCH
RESEARCH IN AGRICULTURAL & APPLIED ECONOMICS

The World's Largest Open Access Agricultural & Applied Economics Digital Library

This document is discoverable and free to researchers across the globe due to the work of AgEcon Search.

Help ensure our sustainability.

Give to AgEcon Search

AgEcon Search
http://ageconsearch.umn.edu
aesearch@umn.edu

*Papers downloaded from **AgEcon Search** may be used for non-commercial purposes and personal study only. No other use, including posting to another Internet site, is permitted without permission from the copyright owner (not AgEcon Search), or as allowed under the provisions of Fair Use, U.S. Copyright Act, Title 17 U.S.C.*

No endorsement of AgEcon Search or its fundraising activities by the author(s) of the following work or their employer(s) is intended or implied.

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

A new model of mechanism design with a boundedly rational agent is studied. A speaker presents a request to a listener who would like to accept the request only if certain conditions are met by the speaker's true case. This persuasion situation is modeled as a leader-follower relationship. The listener first announces and commits to a persuasion rule, i.e. a set of conditions to be satisfied by the case in order for him to be persuaded. Then, the speaker presents a case, though not necessarily the true one. The speaker is boundedly rational in the sense that his ability to come up with a persuasive case is limited and depends on the true case and on the persuasion rule and the way in which it is framed. We fully characterize the circumstances under which the listener's goal can be achieved.

First version: August 2011
Revision: 1 February 2012

The second author acknowledges financial support from ERC grant 269143.
We thank Noga Alon, Ayala Arad, Sambuddha Ghosh, Bart Lipman, Michael Richter, Rani Spiegler and Jaber Zarezadeh. Our special thanks to Chuck Wilson for his very useful comments and suggestions especially regarding Proposition 6.

1. Introduction

"I went to a bar and was told it was full. I asked the bar hostess by what time one should arrive in order to get in. She said by 12 PM and that once the bar is full you can only get in if you are meeting a friend who is already inside. So I lied that my friend was already inside. Without having been 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 in. 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:

As part of a marketing campaign, you have been offered the chance to participate in a lottery. The winner of the lottery will be awarded one million dollars. In order to be eligible to participate, you need to answer the following three questions:

1) Do you usually go to bed before or after midnight?

2) Which of the following do you prefer: cheese cake or chocolate cake?

3) Were you born on an odd or even day of the month?

Your answers must fulfill the following conditions:

R1: If you usually go to bed before midnight and you prefer chocolate cake, then you must have been born on an even day of the month.

R2: If you prefer chocolate cake and you were born on an odd day of the month, then you must usually go to bed before midnight.

R3: If you usually go to bed after midnight and you prefer cheese cake
then you must have been born on an odd day of the month.

R4: If you usually go to bed after midnight and you prefer chocolate cake,
then you must have been born on an odd day of the month.

R5: If you prefer cheese cake and you were born on an even day of the month,
then you must ussually go to bed after midnight .

R6: If you usually go to bed before midnight and you were born on an even day
of the month,
then you must prefer cheese cake.

Consider three individuals: Alice, Bob and Carol.

Alice usually goes to bed before midnight, prefers cheese cake and was born on an even

day of the month. Therefore, she satisfies all six conditions.

Bob's case is different: he goes to bed before midnight, prefers cheese cake and was born on an even day of the month. Bob is not eligible to participate in the lottery since he satisfies the antecedent of R5 but violates its consequent. However, R5 also guides Bob how to modify his case in order to become eligible. By changing his profile to "usually goes to bed after midnight", Bob will satisfy all six conditions and become eligible.

Carol goes to bed before midnight, prefers chocolate cake and was born on an odd day. The only antecedent she satisfies is that of R1. However, if she lies in order to also satisfy the consequent of R1 and states that she was born on an even day of the month, she will violate R6.

Thus, all Alice has to do in order to be eligible for the lottery is to tell the truth. Bob and Carol, on the other hand, have to lie in order to be eligible. However, the codex guides Bob, but not Carol, how to lie successfully.

In this paper, we refer to a set of conditions (rules), such as the one above, as a "codex" and assume that individuals essentially apply the following procedure to come up with a "persuasive case", i.e. a case that satisfies all of a codex' rules:

Step 1. Examine whether your true case satisfies the codex. If it does, then declare the true case. If not, go to Step 2.

Step 2. Find a rule that is violated by your true case (i.e., your true case satisfies its antecedent but not its consequent) and was not examined in a previous iteration of Step 2.

If there are none, go to Step 3.

Otherwise, change your case to one that satisfies the rule and check whether the modified case satisfies the codex.

If it does, then declare the new case. If not, iterate step 2.

Step 3. Declare your true case.

Following such a procedure will make Alice and Bob, but not Carol, eligible for the lottery.

Both of these are examples of a persuasion situation, which typically 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 his desired action regardless of the true case, whereas the listener wishes to be persuaded only in certain cases. In his attempt to persuade the listener, the speaker presents a "case", though not necessarily the true one. However, cheating effectively (i.e., finding a persuasive false case) can be difficult, since it requires the speaker to invent a fictitious profile. Finding a "perfect lie" may require complex calculations, analogous to those required to solve a system of equations. The listener is aware that the speaker may be providing false information but cannot verify this one way or another. The listener is also aware of the speaker's difficulty in coming up with a persuasive false case.

A persuasion situation can be modeled as a leader-follower relationship. First, the listener (leader) publicly announces and commits to a persuasion rule, i.e. a set of conditions that the case presented by the speaker must satisfy in order for the listener to be persuaded. Then, the speaker (the follower) chooses a case to present. In order to persuade the listener, the speaker can present a false case. This is where bounded rationality enters into the analysis. We will assume that the speaker's ability to find a persuasive case is limited and depends on his true case and on the persuasion rule and the way in which it is framed.

Formally, let S be a set of possible 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, if he knew the speaker's true case, would grant the speaker's request. The residual set, $R = S - A$, consists of all the cases in which the listener would reject the speaker's request, if he knew the true case. The speaker, on the other hand, would like the listener to accept his request regardless of the truth. The speaker knows his true case whereas the listener can only rely on the speaker's statement making his decision.

In Glazer and Rubinstein (2004, 2006) the speaker's difficulties in cheating were modeled by introducing a function M, where $M(s) \subseteq S$ is the set of cases that the speaker can present when his true case is s. The listener chooses a persuasion rule, modeled as a set $P \subseteq S$, and commits to accepting the speaker's request if and only if the speaker presents a case in P. After adding a probability measure over the set S to the model, the listener's objective is defined to be the design of a persuasion rule that maximizes the probability that he will take the correct action (from his point of view) subject to the constraint that the speaker maximizes the probability that his request is granted.

In this paper, we assume not only that cheating is difficult, but also that the speaker's

ability to cheat effectively depends on the way in which the mechanism is framed. We model a persuasion rule, referred to as a codex, as a set of conditions formulated in a certain language. A case is persuasive if it meets all the conditions in the codex. The ability of the speaker to present a false case depends on his true case, the set of cases that satisfy the codex and the framing of the codex. In this type of situation, the persuasion rule should be complex enough that a speaker whose case is in R will not be able to 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 explore two notions of implementation. First, we characterize conditions on the set A under which the listener's goal is truthfully implementable, in the sense that there exists a codex which enables the speaker to persuade the listener if and only if his true case should be persuasive (i.e. it belongs to A) and the speaker can do it without lying. Second, we characterize conditions under which the listener's goal is implementable, though not necessarily truthfully, in the sense that there exists a codex such that the speaker is able to persuade the listener if and only if his true case belongs to A, though he may need to lie.

2. The Model

The set of cases

Let V be a set of $K \geq 2$ propositional variables denoted by $v_1,..,v_K$. 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. Denote by $s(v)$ the truth value of the variable v in the case s. We will sometimes present a case s as a K-vector $(s_1,..,s_K)$ of 0's and 1's, where $s_k = 1$ means that $s(v_k) = T$ and $s_k = 0$ means that $s(v_k) = F$.

Let S be the set of all cases. 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 v_1 was "being a female" and v_2 was "being a male").

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 a certain 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 in 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 for his request to be accepted. Then, the speaker (who knows the true case) announces a case that may or may not be the true one. The listener is committed to applying 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 does not violate any of 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). For example, for $K = 4$ the proposition $v_4 \wedge -v_1 \rightarrow v_3$ can be a rule in a codex but $v_1 \rightarrow -v_1$ cannot. For any given rule $\varphi = \wedge_{y \in I} \varphi_y \rightarrow \varphi_x$, we denote $a(\varphi) = \wedge_{y \in I} \varphi_y$ (the antecedent of φ) and $z(\varphi) = \varphi_x$ (the consequent of φ). We interpret a rule as a statement made by the listener: "if you are such and such (described in the antecedent), then you must also be such and such (described in the consequent) in order for your request to be accepted."

2) Coherence: The codex does not contain rules that conflict in the following sense: there is no pair of rules with the same variable v in their consequents, such that their antecedents also not conflict, though the two rules must have opposite truth values for v (namely, the consequent is v in one rule and $-v$ in the other). 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, coherence does not only require that the codex not contain the two rules $v_1 \rightarrow v_2$ and $v_1 \rightarrow -v_2$ but also that it not contain the two rules $v_1 \rightarrow v_3$ and $v_2 \rightarrow -v_3$ (i.e., the antecedents do not conflict but the consequents do). In our view, a codex containing these two rules is problematic: a speaker whose true case, s, is such that $s(v_1) = s(v_2) = T$, will rightly complain that the codex imposes on him two conflicting requirements regarding the variable v_3.

In our lottery example (appeared in the introduction), where the three variables are v_1 ="goes to bed before midnight", v_2 = "prefers cheese cake over chocolate cake" and

v_3="were born on an odd day", the codex presented in the Introduction, consists of the following six rules:

$v_1 \wedge -v_2 \rightarrow -v_3$, $-v_2 \wedge v_3 \rightarrow v_1$, $-v_1 \wedge v_2 \rightarrow v_3$, $-v_1 \wedge -v_2 \rightarrow v_3$, $v_2 \wedge v_3 \rightarrow -v_1$ and $v_1 \wedge -v_3 \rightarrow v_2$.

Notation

For a proposition ψ and a case s, we use the notation $s \models \psi$ to represent the statement "proposition ψ is true in case s". (In other words , $s \models \wedge_{y \in I} \psi_y \rightarrow \psi_x$ unless:

(i) the antecedent of ψ is satisfied, i.e. for all $y \in I$ we have $s(y) = T$ if $\psi_y = y$ and $s(y) = F$ if $\psi_y = -y$; and

(ii) the consequent of ψ is violated, i.e. either $s(y) = T$ and $\psi_y = -y$ or $s(y) = F$ and $\psi_y = y$.

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. $T(\Lambda) = \{s|\ s \models \varphi \text{ for all } \varphi \in \Lambda\} = \cap_{\varphi \in \Lambda} T(\varphi)$.

Guidance

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 regardless of what the true case is. We assume, however, 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 is "guided to" from the true case by using the codex. We say that, given Λ**, the speaker is guided to** 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 $s' \models a(\varphi)$; and

(2) either $z(\varphi) = v$ and $s'(v) = T$ or $z(\varphi) = -v$ and $s'(v) = F$ (and thus $s' \models \varphi$).

In other words, the speaker is guided from s to s' if he can justify to himself any switch from $s(v)$ to $s'(v)$ by a rule whose antecedent refers only to the unchanged variables and is satisfied at s, which suggests that the value of the variable v should be $s'(v)$. We refer to the relation $\rightarrow_\Lambda$ as the guidance relation induced by Λ.

The speaker may be guided from one case to several others cases. For example, suppose that $K = 4$ and Λ contains the three rules $v_1 \rightarrow -v_3$, $v_2 \rightarrow -v_4$ and $v_2 \wedge v_3 \wedge v_4 \rightarrow -v_1$. Then, the speaker is guided by Λ from $(1,1,1,1)$ to any one of the

cases $(1,1,1,1)$, $(1,1,0,1)$, $(1,1,1,0)$, $(1,1,0,0)$ and $(0,1,1,1)$.

The following simple lemma states some of the properties of the relation $\rightarrow_\Lambda$:

Lemma 1:

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

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

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

Given a binary relation $\rightarrow$, define $T(\rightarrow) = \{s \mid \text{for no } t \neq s, \ s \rightarrow t\}$ and $P(\rightarrow) = \{s \mid \text{there is } t \in T(\rightarrow) \text{ such that } s \rightarrow t\}$. If $\rightarrow$ is reflexive, then $T(\rightarrow) \subseteq P(\rightarrow)$.

The following lemma shows that the guidance relation $\rightarrow_\Lambda$ fully conveys the information about $T(\Lambda)$, the set of cases that satisfy the codex Λ.

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

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

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

Persuasion

Given a codex Λ, we say that the speaker in s *can persuade* the listener if $s \rightarrow_\Lambda s'$ for some $s' \in T(\Lambda)$. Define $P(\Lambda) = P(\rightarrow_\Lambda)$. $P(\Lambda)$ is the set of cases in which the speaker can persuade the listener. Note that it is possible for the speaker to be guided from the true case to some cases which are persuasive and some others not. By our definition, it is sufficient for the speaker to be guided to one persuasive case in order to be able to persuade the listener. Note also that we do not allow the speaker to be guided sequentially, i.e., first from s to s' and then from s' to s''.

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 "alter 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.

Note that the "revelation principle" does not hold in our framework and, 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. A useful property of the relation N is that two neighbors of the same case are not neighbors. We will refer to $d(s,s') = |\{v \ | s(v) \neq s'(v)\}|$ 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. 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. 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 $rNsNt$ then there is a unique u such that (r,s,t,u) is a cycle. Denote this u by $N(r,s,t)$.

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 contains 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 $\varphi(s,s')$ be the complete rule $\varphi = \wedge_{y\in V-\{x\}}\ \varphi_y \rightarrow \varphi_x$ where both $s \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge -\varphi_x$ and $s' \vDash \wedge_{y\in V-\{x\}}\ \varphi_y \wedge \varphi_x$. Thus, $s \rightarrow_\Lambda s'$ for any codex

Λ that contains φ.

Canonical codexes

We now define a special form of a codex which does not necessarily have a natural interpretation but is analytically useful.

A codex is canonical if:

(i) It consists of complete rules.

(ii) For every s, there is at most one t such that $s \rightarrow_{\Lambda} t$.

(iii) For every $s \in P(\Lambda) - T(\Lambda)$, there is $r \in R$ such that $r \rightarrow_{\Lambda} s$.

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. Given our assumptions about the speaker's limited ability to come up with a persuasive case, a canonical codex makes the speaker's task relatively simple, since he can simply check the K neighboring cases.

Note that the language of a codex allows the specification of any subset $X \subseteq S$:

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

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

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 $\varphi(s^l, s^{l+1})$ 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 that are satisfied by the same A. However, the codexes differ in the sets of cases in which the speaker can persuade the listener. Only the last one (truthfully) implements A. For notational convenience we identify v_4 with v_1 and v_5 with v_2.

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. that 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 $(0,0,0) \in P(\Lambda)$ although $(0,0,0) \notin A$, 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, as is shown in example 3 for $K = 3$, its complement is not implementable) by the codex Λ that consists of the $K(K-2)$ rules $v_i \rightarrow v_j$ where $j \neq i+1$ ($K+1$ is taken to be 1). Obviously, $T(\Lambda) = \{all\ F, all\ T\}$. The codex guides the speaker to "*all T*" from every case in A except for "*all F*". For any $s \in R$ where there is a unique v_i for which $s(v_i) = T$, the speaker is guided from s only to cases in which v_{i+1} receives the value F and hence they violate the codex. Thus, $s \notin P(\Lambda)$.

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 is guided to slightly exaggerate and to 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 that each consist of two opposing cases. It is sufficient w.l.o.g. to consider the following sets:

(a) $A = \{(1,0),(1,1))\}$ is implementable by $\Lambda = \{-v_1 \to -v_2,\ -v_2 \to v_1,\ v_1 \to v_2\}$, a codex that induces the guidance relation $(0,1) \to_\Lambda (0,0) \to_\Lambda (1,0) \to_\Lambda (1,1)$. Clearly, $T(\Lambda) = \{(1,1)\}$ and $P(\Lambda) = A$.

(b) $A = S - \{(0,0)\})$ is implementable by the codex $\Lambda = \{-v_1 \to v_2,\ v_2 \to v_1\}$ which induces the guidance relation $(0,0) \to_\Lambda (0,1) \to_\Lambda (1,1)$. 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.) $-v_1 \to v_2$ is in the codex. The case $(0,1)$ has to be excluded. Since Λ is coherent it must be that $-v_1 \to -v_2$ is not in the codex and hence $v_2 \to v_1$ is. 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 the speaker cannot be guided from one to 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. In particular, we show that when a set A is truthfully implementable, implementation can be achieved by using a codex that devotes one rule to every case s in R and "misguides" the speaker whose true case is s to a neighboring case in R.

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, $T(\Lambda) = T(\rightarrow_\Lambda)$ and thus for every $s \in R$ there is a case $t \neq s$ such that $s \rightarrow_\Lambda t$. Let $n(s)$ be a neighbor of s that is between s and t. By Lemma 1, we have $s \rightarrow_\Lambda n(s) \rightarrow_\Lambda t$ and therefore $n(s) \notin T(\Lambda)$. The canonical codex $\Lambda' = \{\varphi(s, n(s)) \mid s \in R\}$ 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 $s \rightarrow_\Lambda s'$. Let s_1 be an arbitrary element in R. Then, there exists $s_2 \in R$ such that $s_1 N s_2$ and $s_1 \rightarrow_\Lambda s_2$. Continuing in this manner, we obtain $s_1 N s_2 N \ldots N s_L$ where $s_L = s_{L'}$ for some $L' < L$ and $s_l \in R$ for all 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 ($L + 1$ is taken to be 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 t_N is in the cycle and add $t_1 \rightarrow t_2$ to the relation. Let $\Lambda = \{\varphi(s, s') \mid s \rightarrow s'\}$. Obviously, the relation $\rightarrow$ is anti-symmetric and thus Λ is coherent. The relation $\rightarrow_\Lambda$ is identical to $\rightarrow$ and $P(\Lambda) = T(\Lambda) = A$.

The following proposition describes families of 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 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" with 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 particular variable whose value 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 $s \in R$ is connected by a path in R to a cycle in R.

(1) First, we show that the set R is connected. It is wellknown 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 x such that $s_2 N x N s_4$. The case x must be in A since otherwise (s_2, s_3, s_4, x) forms a cycle in R. The case x 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 x 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) Since $A \subseteq T(v)$ the set $T(-v) \subseteq R$ and it has a counting 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 $s \in R$ belongs to a cycle consisting of the four cases in the set $\{t|\ t(v) = s(v)$ for any $v \notin \{v', v''\}\}$. By assumption these four cases are in R.

An alternative interpretation of truthful implementation: Let $K = 3$ and let $\Lambda = \{v_1 \to v_2,\ v_2 \to v_3\}$. Then, $(1,0,0) \to_\Lambda (1,1,0)$ and $(1,1,0) \to_\Lambda (1,1,1)$. However, by our assumptions, the speaker cannot be guided "sequentially" and thus is not guided from $(1,0,0)$ to the persuasive case $(1,1,1)$. Of course, sequential guidance is not equivalent to full rationality Had we allowed the speaker to be guided sequentially, the following alternative definition of implementation would have applied:

We say that A is implementable in the alternative sense if there exists a codex Λ such that:

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

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

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

Proof: If A is truthfully implementable, then there exists a codex Λ such that $P(\Lambda) = T(\Lambda) = A$. 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 if there exists $s \in R$ and a chain $s = s_1 \to_\Lambda s_2 \ldots \to_\Lambda s_L$ where $s_L \in T(\Lambda)$, then there exists some l for which $s_l \in R$, $s_{l+1} \in T(\Lambda)$ and $s_l \to_\Lambda s_{l+1}$, contradicting the assumption that Λ implements A. Hence, A is implementable in the alternative sense.

On the other hand, assume that Λ implements the set A in the alternative sense. By (ii), there is no member of R in $T(\Lambda)$ and thus by Lemma 2 for any $s \in R$ there exists some s' such that $s \to_\Lambda s'$ and by Lemma 3 we can assume w.l.o.g. that $s' N s$. Had s' been in A, then by (i) there would have been a chain $s' = s_1 \to_\Lambda s_2 \ldots \to_\Lambda s_L$ with $s_L \in T(\Lambda)$ and then we would have $s \to_\Lambda s_1 \to_\Lambda s_2 \ldots \to_\Lambda s_L$, contradicting (ii). Thus, $s' \in R$. Consider the codex $\Lambda' = \{\varphi(s,s') \mid s \in R\}$. Then, $P(\Lambda') = T(\Lambda') = A$.

4. Implementability (not necessarily truthful)

In the following propositions we show that when a set A is implementable (but not necessarily truthfully), implementation can be achieved by using a codex such that:

(i) Some cases in A satisfy the codex.

(ii) For some other cases in A agents are guided by the codex to cheat effectively, i.e. to pretend to be a (neighboring) case in A that satisfies the codex.

(iii) For each case in R, there is a unique rule that "misguides" an agent to a neighboring

case which is either in R or is one of those cases in A in which the agent is guided to cheat effectively.

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

(1) Anti-symmety.

(2) $P(\rightarrow) = A$.

(3) If $s \rightarrow s'$ and $s \neq s'$, then sNs'.

(4) for every s there is at most one s' such that $s \rightarrow s'$.

(5) for every $s \in P(\rightarrow) - T(\rightarrow)$, there is $t \in S - P(\rightarrow)$ such that $t \rightarrow s$.

Proof: Assume that A is implementable by a canonical codex Λ. The relation $\rightarrow_\Lambda$ satisfies properties (1,2,3,4,5)since the coherence of the codex implies (1), the implementability of A by the codex is equivalent to (2), and the fact that the codex is canonical implies (3,4,5).

On the other hand, given a relation $\rightarrow$ that satisfies (1,2,3,4,5), consider $\Lambda = \{\varphi(s,s') \mid s \neq s' \text{ and } s \rightarrow s'\}$. (1) implies that the codex is coherent and (3,4,5) implies that the codex is canonical. The relation $\rightarrow_\Lambda$ is equal to $\rightarrow$ and by (2) we have $P(\Lambda) = P(\rightarrow_\Lambda) = P(\rightarrow) = A$.

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 start with the relation $\rightarrow_\Lambda$ and modify it to become a relation satsifying the five properties spelled out in Proposition 4.

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

(6) Betweenness: If $s \rightarrow_\Lambda s'$ and t is a case "between" s and s', then $s \rightarrow_\Lambda t \rightarrow_\Lambda s'$.

First, define a new reflexive relation $\rightarrow$ as follows:

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

(b) For every $s \in R$, choose one case $s' \neq s$ for which $s \rightarrow_\Lambda s'$. Since $\rightarrow_\Lambda$ satisfies (6), we can assume that $s'Ns$. Since $s \notin P(\Lambda)$, $s' \notin T(\Lambda)$. Define $s \rightarrow s'$.

The relation $\rightarrow$ satisfies (1,2,4) and:

(7) If $s \in R$ then there is a unique s' such that $s \to s'$ and $s' \notin T(\to)$ and $s'Ns$. If $s \in A$ and $s \to s'$, then $s' \in T(\to)$ and all cases between s and s' are in A.

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

(i) For every $s \in A - T(\to)$ such that the set $N(s) \cap T(\to) \neq \emptyset$ and $s \to x$ for $x \notin N(s)$, divert the relation from $s \to x$ to $s \to y$ for some $y \in N(s) \cap T(\to)$.

(ii) Let $s \in A$ be such that $s \to s'$ and $s' \notin N(s)$. Let s'' be a neighbor of s between s and s'. By (7) $s'' \in A$ and by (2) 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 $r \to s''$, then $r \in R$ and by (7), s'' and r are neighbors. Both s and r are neighbors of s'' and let $t = N(s, s'', r)$ (the other joint neighbor of 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' can be r!) and add $r \to t$ and $t \to s$. The new relation satisfies (1), (2), (4) and (7) but with one less element in A which goes to a non-neighbor.

Go back to (i). Following a finite number of iterations we obtain a relation satisfying (1,2,3,4).

Finally, for every $s \in A$ for which $s \to t$ and since there is no $r \to s$ for some $r \in R$, we can omit the arrow $s \to t$ to obtain a relation which also satisfies also 5.

Proposition 6: The set A is implementable if and only if every connected component of R contains (i) a cycle or (ii) a case r such that there are two cases $s, t \in A$ and $rNsNt$.

Proof: Assume that A is implementable. By Proposition 5, it is implementable by a canonical codex Λ and by Proposition 4 there is a binary relation $\to$ satisfying (1,2,3,4,5). Consider a connected component Y of R. By (2,3), for every $r \in Y$ there is $s(r) \in N(r)$ such that $r \to s(r)$. If for every $r \in Y$ the case $s(r) \in R$, then Y must contain a cycle. Otherwise, there is an $r \in Y$ with $r \to s$ and $s \in A$. Then, by (2), it must be that $s \in P(\to) - T(\to)$ and thus there must be some $t \in T(\to) \subseteq A$ such that $s \to t$ and by (3) $rNsNt$.

In the other direction, let $Y_1, .., Y_N$ be a sequence of all connected components of R. If $N = 0$, the set $A = S$ is trivially implementable (see Example 1). If $N > 0$, we inductively construct a relation $\to$ which at the end of stage $n - 1$ will satsify (1,3,4,5) and also $P(\to) = S - Y_1 \cup \ldots \cup Y_{n-1}$ as well as $P(\to) - T(\to) \subseteq A$ (and thus $Y_n \cup \ldots \cup Y_N \subseteq T(\to)$). At the end of stage $n = N$, we obtain a relation satisfying (1,2,3,4,5) and by Proposition 4

the set A is implementable.

We now describe the n'th stage of the inductive construction of $\rightarrow$:

(i) The modification of $\rightarrow$ for the case in which Y_n contains a cycle is straightforward (following the construction in Proposition 2).

(ii) If there exists $r \in Y_n$ which is a neighbor of $s \in P(\rightarrow) - T(\rightarrow)$, then we can extend the relation $\rightarrow$ by adding $r \rightarrow s$ and $\{x \rightarrow y \mid x \in Y_N$ and y is a neighbor of x on the path from x to $r\}$ (there is only one path from x to r since Y_n does not contain a cycle).

We can now concentrate on the case in which there is $r^* \in Y_n$ such that $r^*Ns^*Nt^*$ and $s^*, t^* \in A$ and there is no $r \in Y_n$ that has a neighbor $s \in P(\rightarrow) - T(\rightarrow)$.

(iii) Next, it can be assumed that we can assume that there is no s such that $s \rightarrow s^*$.

If there is and $s \rightarrow s^*$, then $s \notin R$ since if $s \in R$ it must be that $s^* \in P(\rightarrow) - T(\rightarrow)$, a situation already included in (ii). Thus, assume that $s \rightarrow s^*$ and $s \in A$. By (5), there is $r \in R$ such that $r \rightarrow s$. The case $x = N(r^*, s^*, s) \notin R$ since if $x \in R$ it is also in Y_n and xNs and we are back to the situation dealt with in (ii). Also, $x \notin P(\rightarrow) - T(\rightarrow)$ since r^*Nx. Thus, $x \in T(\rightarrow)$ and we can delete $s \rightarrow s^*$ and add $s \rightarrow x$.

(iv) We are left with the situation in which $r^*Ns^*Nt^*$, $s^*, t^* \in A$, $s^* \in T(\rightarrow)$ and there is no $s \rightarrow s^*$.

If s^* has a neighbor x in $A \cap T(\rightarrow)$ then we can extend the relation $\rightarrow$ such that $r^* \rightarrow s^* \rightarrow x$ and for any other $r \in Y_N$ we can add $r \rightarrow s$ where $(r, s, \ldots, r^*)$ is the path from r to r^* in Y_N.

Otherwise, $t^* \notin T(\rightarrow)$ and by (5) there are some cases in R which are directed to t^*.

For every r such that $r \rightarrow t^*$, let $x(r) = N(r, t^*, s^*)$. We have already dealt with the case in which for at least one r we have $x(r) \in A \cap T(\rightarrow)$. We are left with two possibilities to consider:

(a) If $x(r) \in P(\rightarrow) - T(\rightarrow)$, i.e. there is $y \in A$ such that $x(r) \rightarrow y$, we can redirect $r \rightarrow x(r)$.

(b) If $x(r) \in R$ it must be in $Y_1 \cup \ldots \cup Y_{n-1}$ since $x(r)Nr$ and $r \in Y_1 \cup \ldots \cup Y_{n-1}$. Then, for each such r redirect $r \rightarrow x(r)$ and $x(r) \rightarrow s^*$.

There are now remaing cases directed to t^* and as before we can extend the relation such that $r^* \rightarrow s^* \rightarrow t^*$ and $\{r \rightarrow s \mid r \in Y_n$ and $(r, s, \ldots, r^*)$ is the path from r to r^* in $Y_N\}$.

Using the above characterization, Proposition 7 presents families of sets that are

implementable:

Proposition 7: 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 truthfully implementable set and every case in $A - B$ is a neighbor of a case in B.

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

(4) A's connected components are all non-singletons.

(5) A is monotonic in the following sense: if $s \in A$ and s' is a case such that, for every variable v, whenever $s(v) = T$ also $s'(v) = T$, then $s' \in A$.

Proof:

(1) Every case in R is a neighbor of a case in $T(v)$, which has another neighbor in $T(v)$.

(**2)** Since B is truthfully implementable, then by Proposition 1 there is a binary relation $\rightarrow$ such that for any $r \in S - B$ there is a unique $s \in S - B$ such that $r \rightarrow s$ and rNs. Consider any $r \in R$. If the path of the relation $\rightarrow$ that starts from r does not have a cycle in R, then it reaches an element in $A - B$ which has a neighbor in B. Thus, the connected component of R, which contains r, either contains a cycle or includes a case which has a neighbor in $A - B \subseteq A$, which in turn has a neighbor in $B \subseteq A$.

(3) By $|R| \leq K$, any $r \in R$ has a neighbor s in A and if s does not have K neighbors in R then it must have a neighbor in A. If there exists $s^* \in A$ such that $R = N(s^*)$, then for every $r \in R$ there is a ray $(s^*, r, n(r), n^2(r))$ and $n(r)$ and $n^2(r)$ must be in A.

(4) One of the cases in any connected component of R must have a neighbor in A that has a neighbor in A.

(5) The case $A = \{alltruth\}$ is dealt with in Proposition 3(1). Otherwise A is a connected set (all cases are connected to *alltruth*) which is not a singleton.

Corollary: (i) If there exists $s^* \in R$ such that $A \supseteq N(s^*)$ and for any $x \in N(s^*)$ we have $N(x) \subseteq R$, then A is not implementable.

(ii) If all connected components of A are singletons and A is not truthfully

implementable, then A is not implementable.

The set $A = \{(0,1,0,0),(0,0,1,0),(0,0,0,1),(1,1,1,0),(1,1,0,1),(1,0,1,1)\}$ is an example of a set satisfying (2) but not (1).

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 consists of at most two cases or (ii) A is a subset of $T(v)$ or $T(-v)$ for some variable v.

Proof:

(a) By Proposition 3(1), any set A such that $|A| \leq 2$ is truthfully implementable. If $A \supset C$, a set of two opposing cases, then any member of A is a neighbor of a case in C and, thus by Proposition 7(2), is implementable. Similarly, if $A \supset C$, a set of two neighboring cases, and A does not contain a pair of opposing cases, then any case in A is a neighbor of a case in C and, again, by Proposition 7(2), A is thus implementable. We are left with the possibility that A contains 3 or 4 isolated cases. In that case, there is a case $r \in R$ such that all of its neighbors are in A and isolated and therefore by the previous corollary it is not implementable.

(b) If A satsifies (i) or (ii), it is truthfully implementable by parts (1) and (3) of Proposition 3.

On the other hand, if A is truthfully implementable and contains more than 2 cases, then by Proposition 2 the set R contains a cycle of 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 $T(-v_1)$.

5. Discussion

5.1. 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 belief, we conducted a series of experiments based on the lottery example described in the Introduction. Subjects from more than 30 countries who had all taken a game theory course were asked to participate in a short web-based experiment. The subjects were first asked three questions about themselves:

1) On most days, do you go to bed **before** midnight or **after** midnight?

2) Which of the following do you prefer: **cheese** cake or **chocolate** cake?

3) Were you born on an **odd** or **even** day of the month?

After answering the three questions, the subjects were presented with a new screen:

"Assume now that as part of a marketing campaign you have been offered the chance to participate in a lottery. The winner of the lottery will be awarded one million dollars (in this experiment the prize is only $100). In order to be eligible to participate, you must answer three questions about yourself and your answers must not violate any of the following six restrictions:"

Then the codex $\Lambda_1 = \{R1, \ldots, R6\}$ described in the Introduction was presented in random order and the subjects were asked the following question:

"Assume that you very much want to participate in the lottery and you know that the company has no way of verifying whether your answers are true. How would you answer the following three questions in this case?"

This was followed by the same three questions presented above.

Letting v_1 ="before", v_2 = "cheese" and v_3="odd", the codex Λ_1 consists of the six rules: $v_1 \wedge -v_2 \rightarrow -v_3$, $-v_2 \wedge v_3 \rightarrow v_1$, $-v_1 \wedge v_2 \rightarrow v_3$, $-v_1 \wedge -v_2 \rightarrow v_3$, $v_2 \wedge v_3 \rightarrow -v_1$ and $v_1 \wedge -v_3 \rightarrow v_2$. The induced guidance relation is: $111 \rightarrow_{\Lambda_1} 011$, $100 \rightarrow_{\Lambda_1} 110$, $010 \rightarrow_{\Lambda_1} 011$, $101 \rightarrow_{\Lambda_1} 100$, $001 \rightarrow_{\Lambda_1} 101$ and $000 \rightarrow_{\Lambda_1} 001$. Thus, $T(\Lambda_1) = \{011, 110\}$ and $P(\Lambda_1) = T(\Lambda_1) \cup \{111, 100, 010\}$.

We partitioned the subjects into three groups $T = T(\Lambda_1)$, $P = P(\Lambda_1) - T(\Lambda_1)$ and $R = R(\Lambda_1)$, according to their "declared profile" in the first screen. Each row in the table refers to one of these groups. The numbers in the first column, denoted by *T*, represent the proportion of each group whose answers in the second screen belong to $T(\Lambda_1)$. The numbers in the second column, denoted by "Honest", represent the proportion of each group who submitted the same profile in the second screen as in the first. (Notice that the answer of a subject in *T* will be counted in two columns.) The numbers in the third column, denoted by "Other", represent the proportion of each group whose answer was neither in *T* nor honest.

Λ_1	*T*	*MRT*	**Honest**	**Other**	**N**
T	80%	125*s*	71%	20%	104
P	54%	157*s*	29%	17%	180
R	36%	317*s*	34%	30%	261

Following are our main observations:

1) The results support our main assumption that the ability of a subject to come up with an eligible case strongly depends on his true case. While 80% of the subjects in *T* presented a persuasive case, the success rate dropped to 54% among the subjects in *P* and to 36% among the subjects in *R*.

2) The median response time of successful subjects varied from 125*s* for subjects in *T* to 157*s* for subjects in *P* and to 317*s* for subjects in *R*. This supports our assumption that subjects in *R* find it more difficult to come up with a persuasive case than subjects in *P* and *T*. Note, however, that the sample is not large enough to draw firm conclusions based on response time.

3) According to Λ_1, each of the three cases in *P* is guided by the codex to a single case in *T* (two cases are guided to 011 and one is guided to 110). Indeed, of the 97 subjects in *P* who presented a persuasive case, 68% followed the guide. This result supports our main assumption that subjects use the codex as a guide in coming up with a persuasive case using their true case as a starting point.

4) The choices of the 251 subjects in $P \cup R$ who did not present a persuasive case are far from being random. 56% of the subjects were honest while 35% chose a case that one of the six rules led to (100, 101, or 001). Only 9% chose a case that none of the six rules led to (111, 010, or 000).

5) One could suggest an alternative model of bounded rationality, according to which a subject is restricted to presenting either the true case or one of its three neighboring cases (this model is in line with Glazer and Rubinstein (2006)). However, the results do not support this hypothesis. First, note that for the 111 and 010 subjects the two persuasive cases are neighboring ones. However, they are guided by the codex only to 011 (and not to 110). Indeed, 75% of the 72 subjects who presented a persuasive case followed the guide and chose 011. Second, the success rate of the 001 subjects (37%) who had a neighboring case in *T* was no different than that of the other two *R* cases (101 and 000) which do not

have a neighboring case in *T* (37% and 33% respectively).

An alternative explanation for the popularity of 011 among the 111 and 010 subjects is that 011 is confirmed by two rules. Therefore, we conducted a second experiment with a modified codex Λ_2 whose guidance relation is: $111 \rightarrow_{\Lambda 2} 011$, $100 \rightarrow_{\Lambda_2} 110$, $010 \rightarrow_{\Lambda_2} 110$, $001 \rightarrow_{\Lambda 2} 011$, $101 \rightarrow_{\Lambda_2} 100$ and $000 \rightarrow_{\Lambda_2} 001$. For this codex, $T(\Lambda_2) = T(\Lambda_1)$ but $P(\Lambda_2) - T(\Lambda_2)$ consists of four cases: 111 and 011 (guided by the codex to 011), and 100 and 010 (guided to 110). The following table summarizes some of the results:

Λ_2	**T**	**Honest**	**Other**	**N**
T	88%	75%	12%	52
P	63%	27%	10%	123
R	45%	15%	40%	65

Once again, we observe a strong dependence of the success rate on the subject's true case. Thus, almost all *T* cases, 63% of the *P* cases and only 45% of the *R* cases came up with a persuasive case.

Particularly interesting is the group of 123 subjects whose case is in *P*. Each of the four cases in *P* is guided by the codex to a unique case in *T*. Of the 78 successful subjects in *P*, 51 subjects (65%) seem to be guided by the codex guidance. We believe that this strongly supports our main assumption that individuals first examine whether their true case satisfies the codex and if it is not they are guided by the codex to the next case for consideration.

Finally we also experimented with another codex, Λ_3, which truthfully implements $\{110, 011\}$. The induced guidance relation is $111 \rightarrow_{\Lambda_3} 101$ $100 \rightarrow_{\Lambda_3} 101$ $010 \rightarrow_{\Lambda_3} 000$, $101 \rightarrow_{\Lambda_3} 100$ $001 \rightarrow_{\Lambda_3} 101$ and $000 \rightarrow_{\Lambda_3} 001$. The following table summarizes the results.

Λ_3	**T**	**Honest**	**Other**	**N**
T	81%	77%	19%	26
R	34%	44%	22%	100

Once again, there is a dramatic difference between the success rates of the *T*'s (81%) and the *R*'s (34%). The success rate (34%) and the median response time (332s) of the *R*'s are similar to those of the *R*'s in the previous experiments and only one *R* subject chose a

case that none of the rules led to.

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 proves the NP completeness of deciding whether a given Boolean formula in conjunctive normal form has an assignment that makes the formula true.

Kartik (2009) analyzed 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.

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 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 investigated the implementation problem when some of the players are "faulty", in the sense that they fail to act optimally. He introduces a solution concept called "fault-tolerant implementation", which requires robustness to deviations from equilibrium. He 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 guided by the mechanism itself 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 persuade 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 situations.

(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 some undeserving speakers to pretend to be deserving ones whose case is rejected by the codex while the deserving speakers are 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, 157-173.

Glazer, Jacob and Ariel Rubinstein (2004). "On Optimal Rules of Persuasion". *Econometrica*, 72, 1715-1736.

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.

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