EPSILON BAYESIAN IMPLEMENTATION

A Master's Thesis

by
EMRE ERGİN

Department of
Economics
İhsan Doğramacı Bilkent University
Ankara
July 2014

To my family

EPSILON BAYESIAN IMPLEMENTATION

The Graduate School of Economics and Social Sciences
of
İhsan Doğramacı Bilkent University

by

EMRE ERGİN

In Partial Fulfillment of the Requirements For the Degree
of
MASTER OF ARTS

in

THE DEPARTMENT OF
ECONOMICS
İHSAN DOĞRAMACI BİLKENT UNIVERSITY
ANKARA

JULY 2014

I certify that I have read this thesis and have found that it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Arts in Economics.

Assist. Prof. Dr. Nuh Aygün Dalkıran

Supervisor

I certify that I have read this thesis and have found that it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Arts in Economics.

Assist. Prof. Dr. Rahmi İlkılıç

Examining Committee Member

I certify that I have read this thesis and have found that it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Arts in Economics.

Assist. Prof. Dr. Burcu Esmer

Examining Committee Member

Approval of the Graduate School of Economics and Social Sciences

---

Prof. Dr. Erdal Erel

Director

ABSTRACT

EPSILON BAYESIAN IMPLEMENTATION

ERGİN, Emre Ergin

M.A., Department of Economics

Supervisor: Assist. Prof. Nuh Aygün Dalkıran

July 2014

We provide necessary and sufficient conditions for epsilon-Bayesian Implementation. Results of Jackson (1991) are extended upon the environments where the agents has some level of bounded rationality. Yet, his necessity condition, Bayesian Monotonicity is not nested with our necessity condition, epsilon-Bayesian Monotonicity.

Keywords: Bayesian Implementation, Epsilon Equilibrium, Bayesian Monotonicity

ÖZET

EPSİLON BAYEZYEN UYGULAMA

ERGİN, Emre

Yüksek Lisans, Ekonomi Bölümü

Tez Yöneticisi: Yard. Doç Dr. Nuh Aygün Dalkıran

Temmuz 2014

Epsilon-Bayezyen Uygulama için gerek ve yeter koşullar sunuyoruz. Jackson (1991)'in sonuçlarını kişilerin belli bir seviyede irrasyonelliği olduğu durumlara genişletiyoruz. Ancak, onun gerek koşulu, Bayezyen Monotonlukla, bizim gerek koşulumuz epsilon-Bayezyen Monotonluk birbirini gerektirmiyor.

Anahtar Kelimeler: Bayezyen uygulama, Epsilon Denge, Bayezyen monotonluğu.

ACKNOWLEDGEMENTS

I am grateful to Semih Koray and Kemal Yildiz for their invaluable comments. My supervisor, Nuh Aygün Dalkiran helped me to overcome many adversities on my path, and I am indebted to him.

I would like to thank my family for their unconditional love and continuous support. Without them, this study and I would be incomplete.

Finally, but not least, I thank TÜBİTAK “The Scientific & Technological Research Council of Turkey" for the financial support during my study. However, of course, all errors and omissions are mine.

TABLE OF CONTENTS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

After society has agreed upon a social choice rule, there is no certain way to implement that rule correctly, since preferences can only be considered through announcements. Agents may report their preferences falsely, according to a strategy that ensures defined social choice rule benefit them. Our interest in this topic stems from the casual exposure to authority's decision on some alternatives in daily life. For example, who will get which teaching assistantship in an efficient way, when the preferences of each graduate student is unknown? Is there a way to ensure that each student won't manipulate the found system?

Since we don't want to have unwanted strategy profiles as equilibrium, our work deals with the problem of full implementation. Gibbard (1973) and Satterthwaite (1975) showed that there can be no strategy-proof game which is not dictatorial following the work of Arrow (1963). So, after Groves and Ledyard (1977), Hurwicz (1979), and Schmeidler (1980) which include constructing nondictatorial game mechanisms in economic environments, implementation literature highly depends on Nash Equilibrium. Maskin(1999) includes an elegant constructive proof of existence of a mechanism that implements a social choice rule via Nash Equilibrium.

When there is incomplete information, agents must also speculate about

other agents' true preferences when deciding their own announcement. That brings forward Bayesian Equilibrium concept. Postlewaite and Schmeidler (1986) and Palfrey and Srivastava (1987) analyzed incomplete information case in nonexclusive information assumption i.e. with N agents, every group of N-1 agents collectively has complete information. They found that, only Bayesian monotonicity will suffice for that kind of implementation. Palfrey and Srivastava (1989) analyzed exclusive information case, and found that a new condition is necessary for such implementation: Incentive Compatibility, and a stronger version of it is sufficient for full implementation. Jackson (1991) extends their work, and finds necessary and sufficient conditions with possibility of externality and noneconomic environments.

In bounded rationality, there is a possibility that agents will not move according to their best interest. Considering many experimental findings, and the general debate on whether Nash Equilibrium is a good representation on reality, bounded rationality is a rather new, but necessary adding to the implementation literature. So it is crucial to analyze what happens, when the agents not strictly achieve to get their best responses, but rather make a decision that is close to it. Barlo and Dalkiran(2009) extends Maskin (1999) to analyze bounded rationality case, and showed that there can be implementation with modified version of monotonicity and limited veto power (Benoit and Ok, 2006).Barlo and Dalkiran(2014) did a similar extension for Bergemann and Morris (2008). Common crucial result is that, implementable Social Choice Correspondences set and epsilon implementable SCC set are not subset of one another for both extensions. That means, bounded rationality will have different policy implications.

Our paper will provide necessary and sufficent conditions for a social choice set to be implementable via epsilon-Bayesian Equilibrium when agents' preferences can be represented by cardinal utilities in incomplete information environment. That is, we will extend Jackson (1991) to include bounded ra-

tionality as defined in Radner (1980), just as Barlo and Dalkiran (2009) did it with Maskin (1999). We will use a constructive proof to show that Social Choice Functions that provide *epsilon-Incentive Compatibility* and *epsilon-Bayesian Monotonicity*, are epsilon-implementable in economic environments. In proof, we will construct a general message space and a mechanism which will ensure epsilon Bayesian implementability under those conditions.Then our paper includes examples of social choice functions that justifies using epsilon-Bayesian implementation concept.

Chapter 2 describes the model, Chapter 3 provides our main result, Chapter 4 covers two examples of social choice rules to show that our and Jackson's conditions are not nested , Chapter 5 concludes.

CHAPTER 2

PRELIMINARIES

2.1 Preliminaries

Below, we represent the notation we use throughout the paper.

- $N = \{1, 2, ..., n\}$ denotes the set of agents.
- Θ_i denotes the finite set of possible types of agent i.
- ε denotes the level of bounded rationality of agents. (ε is state independent.)
- $\Theta = \Theta_1 \times ... \times \Theta_n$ denotes the possible states of the world. (The knowledge is distributed among the agents.)
- A denotes the set of alternatives which is assumed to be fixed across states.
- $\mathcal{F} = \{f | f : \Theta \to A\}$ denotes the set of all social choice functions.
- $F \subset \mathcal{F}$ is said to be a social choice set.
- $q_i(\theta)$ is the probability measure on the set of possible states.

- $\pi_i(\theta_i) = \{t \in \Theta \mid t_i = \theta_i\}$ is the set of states which i believes may be the true state of the world when his private information is θ_i.

- Each agent i's state dependent preferences over the set of alternatives is given by $u_i : A \times \Theta \to \mathbb{R}_+$.

Definition 1 (Mechanism)**.** A *mechanism*, alternatively a game form, describes a message/strategy space $M_i \neq \emptyset$ for each agent $i \in N$ and specifies an outcome function $o : M \to A$, where $M = \times_{i \in N} M_i$. We denote a normal–form mechanism by $\mu = (M, o)$. In state θ, the mechanism μ together with the preference profile u^θ define a game of incomplete information. In such a game, a strategy for player i is a function $\sigma_i : \Theta_i \to M_i$. A strategy profile is denoted by $\sigma^*(\theta) = \times_{i \in N} \sigma_i(\theta_i)$.

Below is the definition of an Epsilon-Bayesian Equilibrium of a mechanism μ:

Definition 2 (Epsilon-Bayesian Equilibrium)**.** A strategy profile σ^* is called an Epsilon-Bayesian equilibrium of μ, if for all i, θ_i and $\widetilde{\sigma}_i$ we have

$$\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[o(\sigma(\theta)), \theta] \geq \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[o(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})), \theta] - \varepsilon$$

We continue with the definition of Epsilon-Bayesian Implementation.

Definition 3 (Epsilon-Bayesian Implementation)**.** F is said to be *Epsilon-Bayesian implementable* (in pure strategies) if there exists a mechanism $\mu = (M, o)$ such that:

1. For every $f \in F$, there exists an Epsilon-Bayesian Equilibrium σ^* of $\mu = (M, o)$ that satisfies

$$f(\theta) = o(\sigma^*(\theta)) \text{ for all } \theta \in \Theta;$$

2. For every Epsilon-Bayesian Equilibrium σ^* of $\mu = (M, o)$, there exists $f \in F$ such that:

$$o(\sigma^*(\theta)) = f(\theta) \text{ for all } \theta \in \Theta.$$

2.2 The conditions

We list necessary and sufficient conditions for Epsilon-Bayesian Implementation:

Definition 4 (Closure)**.** Recall that $\pi^i(\theta^i) = \{t \mid t^i = \theta^i\}$ and the sets π^i form a partition Π^i over Θ. Let Π denote the common knowledge concatenation defined by $\Pi^1, ..., \Pi^N$. That is, Π is the finest partition which is coarser than each Π^i.

Pick any two disjoint events $\hat{\Theta}$ and $\tilde{\Theta}$ such that $\hat{\Theta} \cup \tilde{\Theta} = \Theta$ and for any $\pi \in \Pi$ either $\pi \subset \hat{\Theta}$ or $\pi \subset \tilde{\Theta}$. (Thus, $\hat{\Theta}$ and $\tilde{\Theta}$ are such that, given any state in Θ, all agents know whether the state lies in $\hat{\Theta}$ or $\tilde{\Theta}$ and this is common knowledge among agents.) A social choice set F satisfies *closure* if for any $e \in F$ and $f \in F$ there exists $h \in F$ such that $g(\theta) = e(\theta)\ \forall \theta \in \hat{\Theta}$ and $g(\theta) = f(\theta)\ \forall \theta \in \tilde{\Theta}$.

This is the same condition in Jackson (1991) and was first discussed by Postlewaite and Schmeidler (1986) and Palfrey and Srivastava (1987).

Definition 5 (Epsilon bounded Incentive Compatibility)**.** A social choice set F satisfies Epsilon bounded Incentive Compatibility (EIC) if for all $f \in F$, i, and $t_i \in \Theta_i$ such that

$$\sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[f(\theta), \theta] \geq \sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[f(t_i, \theta_{-i}), \theta] - \varepsilon\ \forall \theta_i \in \Theta.$$

One can easily see that when $\varepsilon = 0$, EIC condition will coincide with the Incentive Compatibility condition in Jackson(1991).

Definition 6 (Deception)**.** A *deception* by agent $i \in N$ is denoted by $\alpha_i : \Theta_i \to \Theta_i$. A deception α_i by agent i with a true type θ_i is interpreted as i's reported type, $\alpha_i(\theta_i)$, as a function of his true type. A *profile of deceptions* is denoted by $\alpha(\theta) = (\alpha_1(\theta_1), \alpha_2(\theta_2), ..., \alpha_n(\theta_n))$. $f \circ \alpha(\theta) = f(\alpha(\theta))$ and naturally, it defines of a deception's outcome on a single state.

Definition 7 (Epsilon-Bayesian Monotonicity)**.** F is said to be *Epsilon-Bayesian Monotonic* (EBM) if, for every $f \in F$ and deception α with $f \circ \alpha \notin F$, there exists $i \in N, r : \Theta_{-i} \to A$ such that

$$\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[r(\alpha_{-i}(\theta_{-i})), \theta'] > \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[f \circ \alpha(\theta), \theta'] + \varepsilon \tag{2.1}$$

for some $\theta_i' \in \Theta_i$ *and*

$$\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[f(\theta), \theta] \geq \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[r(\theta_{-i}), \theta] - \varepsilon \tag{2.2}$$

for all $\theta_i \in \Theta_i$

Here i can be interpreted as a whistle-blower and r as a reward: Condition (1) guarantees that i has (strict) incentive to blow the whistle when the outcome is incompatible with F and Condition (2) makes sure that if the outcome is compatible with F, i does not have enough incentive to blow the whistle.

CHAPTER 3

MAIN RESULTS

We are working in epsilon-economic environment which is defined below.

The social choice function $f/_{\Psi}h$ is defined along set $\Psi \subset \Theta$ as $[f/_{\Psi}h(\theta)] = f(\theta)\ \forall\theta \in \Psi$ and $[f/_{\Psi}h(\theta)] = h(\theta)$ otherwise.

Notation:

$$fP_i^{\varepsilon}(\theta_i)\tilde{f} \Leftrightarrow \sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[f(\theta),\theta] > \sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[\tilde{f}(\theta),\theta] + \varepsilon$$

Definition 8 (Epsilon-Economic Environment)**.** An environment satisfies (EE) if for any $h \in \mathcal{F}$ and $\theta \in \Theta$, there exist i and j $(i \neq j)$ such that $f \in F$ and $g \in G$ while f and g are constant, $f/_{\Psi}hP_i^{\varepsilon}(\theta_i)h$ and $g/_{\Psi}hP_j^{\varepsilon}(\theta_j)h$ for all $\Psi \subset \Theta$ with $\theta \in \Psi$. Environments satisfying (EE) are said to be epsilon-economic.

Condition (EE) requires that for any given social choice function and state, there are at least two agents who have strict incentives (more than ε) to alter the social choice function. The condition is economic in nature since it implies that agents can not be simultaneously satiated, thus there is no ultimate social choice.

Theorem 1. In an environment which satisfies (EE), $N \geq 3$ a social choice set F is implementable, if and only if F satisfies C, EIC and EBM.

Proof. **Necessity:**($\Rightarrow$)

[Closure:] Let (M, g) implement F. Take any $f, f' \in F$ such that $f \neq f'$. Suppose $\hat{\Theta} \cup \tilde{\Theta} = \Theta$ where for any $\pi \in \Pi$ we either have $\pi \in \hat{\Theta}$ or $\pi \in \tilde{\Theta}$.[1] Consider the corresponding Epsilon Bayesian Equilibria σ^* and σ'^* where $f(\theta) = g(\sigma^*(\theta))$ and $f'(\theta) = g(\sigma'^*(\theta))$ for all $\theta \in \Theta$. Let $\sigma''^*(\theta) = \sigma^*(\theta)$ for all $\theta \in \hat{\Theta}$ and $\sigma''^*(\theta) = \sigma'^*(\theta)$ for all $\theta \in \tilde{\Theta}$. Then $\sigma''^*(\theta)$ must be another Epsilon-Bayesian Equilibrium. Letting $f''(\theta) = f(\theta)$ when $\theta \in \hat{\Theta}$ and $f''(\theta) = f'(\theta)$ when $\theta \in \tilde{\Theta}$, we get $f''(\theta) = g(\sigma''^*(\theta))$ hence by (2) of Epsilon-Bayesian Implementation we must have $f'' \in F$ which assures Closure.

[EIC:] Take any $f \in F$ and the corresponding Epsilon-Bayesian Equilibrium σ such that $g[\sigma(\theta)] = f(\theta)$ for all $\theta \in \Theta$. Consider any $i, t_i \in \Theta_i$ and the strategy $\widetilde{\sigma}_i(\theta_i) = \sigma(t_i)$ for some $t_i \in \Theta_i$. Since σ is an equilibrium we must have:

$$\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[g(\sigma(\theta)), \theta] \geq \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[g(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})), \theta] - \varepsilon$$

Since $h[\sigma(\theta)] = f(\theta)$ for all $\theta \in \Theta$ we have $g((\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})) = g(\sigma(t_i, \theta_{-i})) = f(t_i, \theta_{-i})$ which establishes (EIC).

[EBM:]Let f and σ be as above. Consider a deception α such that there is no $g \in F$ with $g(\theta) = f \circ \alpha(\theta)$ for all $\theta \in \Theta$. We must hence have that $\sigma \circ \alpha$ is not an Epsilon-Bayesian-equilibrium. Therefore, there exist i, θ_i, and $\tilde{m}_i$ such that

$$\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[h(\tilde{m}_i, \sigma_{-i} \circ \alpha_{-i}(\theta_{-i})), \theta] > \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[h(\sigma \circ \alpha(\theta)), \theta] + \varepsilon$$

Since $f(\theta) = h(\sigma(\theta))$ for all $\theta \in \Theta$ we have $g(\sigma \circ \alpha(\theta)) = f \circ \alpha(\theta)$. Defining $r(\theta_{-i}) := h(\tilde{m}_i, \sigma_{-i}(\theta_{-i}))$ follows (1) of EBM:

[1]Recall that $\pi^i(\theta^i) = \{t \mid t^i = \theta^i\}$ and the sets π^i form a partition Π^i over Θ. Let Πdenote the common knowledge concatenation defined by $\Pi^1, ..., \Pi^N$. That is, Π is the finest partition which is coarser than each Π^i.

$$\sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[r(\alpha_{-i}(\theta_{-i})),\theta] > \sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[f\circ\alpha(\theta)),\theta] + \varepsilon$$

On the other hand, σ is an Epsilon-Bayesian Equilibrium implies:

$$\sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[h(\sigma(\theta)),\theta] \geq \sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[h(\tilde{m}_i,\sigma_{-i}(\theta_{-i})),\theta] - \varepsilon$$

Again, since $f(\theta) = h(\sigma(\theta))$ and $r(\theta_{-i}) = h(\tilde{m}_i,\sigma_{-i}(\theta_{-i}))$ follows (2) of EBM as well:

$$\sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[f(\theta),\theta] \geq \sum_{\theta\in\pi_i(\theta_i)} q_i(\theta)u_i[r(\theta_{-i})),\theta] - \varepsilon\,.$$

Sufficiency: ($\Leftarrow$)

Let F be a social choice set which satisfies *closure*, EIC and EBM. Consider the following mechanism:

Define message space of each agent i as: $M_i = \Theta_i \times F \times \{\varnothing \cup \mathcal{F}\} \times \mathbb{N}$ and $M = \times_{i\in N} M_i$.

That is, each agents sends a 4 dimensional message. The first coordinate is chosen form their type space, the second coordinate is a social choice function from the social choice set, F, to be implemented, third coordinate can either be empty or is an (unrestricted) social choice function, $\mathcal{F}$, fourth coordinate is chosen to be a natural numbers.

To define the outcome function we partition M as follows:

$$M^0 = \{m \in M | m_j = (., f, \varnothing, k) \text{ for all } j \in N \text{ for some } f \in F \text{ and } k \in N\}$$

$$M_i^* = \{m \in M | m_j = (., f, \varnothing, k) \text{ for all } j \neq i \text{ for some } f \in F \text{ and } m_i = (., f, \varnothing, l) \text{ or } m_i = (., f', ., .)\}$$

$$M_i^{**} = \{m \in M | m_j = (., f, \varnothing, k) \text{ for all } j \neq i \text{ and } m_i = (., f, \tilde{f}, .) \text{ for some } f \in F\}$$

$$M^{***} = \{m \in M | m \notin M^* \cup M^{**}\}$$

where $M^* = \cup_i M_i^*$ and $M^{**} = \cup_i M_i^{**}$. Clearly, $M = M^0 \cup M^* \cup M^{**} \cup M^{***}$.

For any given message profile m let $\theta^m = m_1^1 \times m_2^1 \times \ldots \times m_n^1$ where m_i^1 is the first coordinate of the message sent by agent i. Consider the outcome function $h : M \to A$ given as below:

- $o(m) = f(\theta^m)$ if $m \in M^0 \cup M^*$;
- $o(m) = \tilde{f}(\theta^m)$ if $m \in M_i^{**}$ and $\sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[f(\theta), \theta] \geq \sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[\tilde{f}(m_i^1, \theta_{-i}), \theta] - \varepsilon$ for all $\theta_i \in \Theta_i$;
- $o(m) = f(\theta^m)$ if $m \in M_i^{**}$ and $\sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[o(m_i^1, \theta_{-i}), \theta] > \sum_{\theta \in \pi_i(\theta_i)} q_i(\theta) u_i[f(\theta), \theta] + \varepsilon$ for some $\theta_i \in \Theta_i$;
- $o(m) = m_{i^*}^3(\theta^m)$ if $m \in M^{***}$ where $m_{i^*}^3$ is the social choice function which is the third coordinate of the message sent by agent $i^* = \text{argmax}_i\{m_i^4\}$, i.e., i^* is the agent who has the highest natural number in his message's fourth coordinate.

Now we will show that $\mu = (M, g)$ as defined above implements F in Epsilon-Bayesian Equilibrium.

Lemma 1. For every $f \in F$, there exists an Epsilon-Bayesian Equilibrium σ^* of μ such that $h(\sigma(\theta)) = f(\theta)$ for all $\theta \in \Theta$.

Proof. Take any $f \in F$ consider σ^* such that $\sigma_i^*(\theta_i) = (\theta_i, f, \varnothing, ., k)$ for all $i \in N$ and for some $k \in \mathbb{N}$. Hence, by construction of μ we have $o[\sigma(\theta)] = f(\theta)$ for all $\theta \in \Theta$ as desired.

To see that σ^* is an equilibrium consider a deviation $\widetilde{\sigma}_i$ by agent i. First observe that we can either have $(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})) \in M_i^*$ or $(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})) \in M_i^{**}$. If $(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})) \in M_i^*$, that is, $\widetilde{\sigma}_i(\theta_i) = (\widetilde{\theta}_i, f, \varnothing, l)$ or $\widetilde{\sigma}_i(\theta_i) = (\widetilde{\theta}_i, f', \hat{f}, l)$ then the outcome changes to $f(\widetilde{\theta}_i, \theta_{-i})$. It follows from EIC that such a deviation cannot be profitable more than ε. If $(\widetilde{\sigma}_i(\theta_i), \sigma_{-i}(\theta_{-i})) \in M_i^{**}$, then $\widetilde{\sigma}_i(\theta_i) = (\widetilde{\theta}_i, f, \tilde{f}, l)$, then the outcome is either $f(\widetilde{\theta}_i, \theta_{-i})$ or $\widetilde{f}(\widetilde{\theta}_i, \theta_{-i})$. In the case of former again it follows from EIC that such a deviation cannot be profitable more than ε. For the case of latter, by construction, we must have

$$\sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[f(\theta), \theta] \geq \sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[\tilde{f}(\widetilde{\theta}_i, \theta_{-i}), \theta] - \varepsilon \text{ for all } \theta_i \in \Theta_i$$

which means such a deviation is not profitable more than ε as well. □

Lemma 2. For every Epsilon-Bayesian Equilibrium σ^* of $\mu = (M, h)$, there exists $f \in F$ such that $h(\sigma^*(\theta)) = f(\theta)$ for all $\theta \in \Theta$.

Proof. Let σ^* be an Epsilon-Bayesian equilibrium and let α describe the announcement of the state (m^1 as a function of θ) under σ .

Suppose that there does not exist a social choice function f in F which is equivalent to $h(\sigma)$. We will find a deviating player to prove by contradiction. Without loss of generality, we can take $f = g \circ \alpha$ since for any f, there is some g and α which satisfies this. [2]

We are interested in finding a deviating player, so we will look for all cases whether if there is such deviating player i. Remembering our mechanism, a

[2] $g = f$ and α being the identity deception is the trivial case.

message profile m, can belong to four different sets, namely, M^0,$M_i^\star$,$M_i^{\star\star}$ and $M^{\star\star\star}$.

Case 1 $\sigma^* \in M^0$

A player i can change the outcome by deviating only with choosing his $\widetilde{m}_i = \left(., f, \tilde{f}, .\right)$ which satisfies

$\sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[f(\theta), \theta] \geq \sum_{\theta \in \pi_i(\theta)} q_i(\theta) u_i[\tilde{f}(m_i^1, \theta_{-i}), \theta] - \varepsilon$ for all $\theta_i \in \Theta_i$.
Any other deviation will put the $\widetilde{\sigma}$ into $M_i^{\star\star}$ with ,$o(m) = f(\theta^m)$ thus providing no possible deviation.

We know that by EBM, since $f = g \circ \alpha \notin F$, there is some i and r satisfies $r(\alpha_{-i}(\theta_{-i}))P_i^\varepsilon(\theta_i)f$ for some $\theta_i \in \Theta_i$, while $gR_i^\varepsilon r(\theta_{-i})$ for all $\theta_i' \in \Theta_i$. It can easily be seen that second condition of EBM coincides with the requirement for $o(m) = \tilde{f}(\theta^m)$ in our mechanism. We know that this deviation will be beneficial for i, because of first condition of EBM. So, for this case, i has profitable deviation.

Case 2 $\sigma^* \in M_i^\star \cup M_i^{\star\star}$

If the starting included a deviation from some player i, then some other player j could change his message with highest natural number as its fourth component, contradicting σ^* being a Epsilon Bayesian Equilibrium. This is due to environment is epsilon-economic, all agents can not be simultaneously satiated.

Case 3 $\sigma^* \in M^{\star\star\star}$

Since the environment is economic, there is no ultimate social choice. So, whenever this is the case, some player j can deviate to his ultimate choice using highest natural number as his fourth component, contradicting σ being a Epsilon Bayesian Equilibrium.

Therefore, i is better off by submitting $(\alpha(\theta), f, r, .)$ whenever θ_i is observed.This contradicts that σ is an equilibrium, so our supposition about nonexistence of a social choice function equivalent to $h(\sigma)$ is wrong. □

□

CHAPTER 4

EXAMPLES

This chapter proves that epsilon-Bayesian Monotonicity (henceforth EBM) and Bayesian Monotonicity (henceforth BM) are not nested. This directly implies implementable social choice rules with or without bounded rationality are not nested. Examples are tested with a C++ code, which is included in the appendix.

4.1 BM does not imply EBM

Following is an example to Bayesian Monotonic social choice rule which is not epsilon-Bayesian monotonic (result holds for $\epsilon = 0.75$).

Suppose N=1,2,3 and $\Theta_i = \{0, 1\}$. Hence a type profile $(\theta_1, \theta_2, \theta_3) \in \Theta = \{0, 1\}^3$. There are 8 possible outcomes given by

$$A = \{\textit{0-0-0}, \textit{0-0-1}, \textit{0-1-0}, \textit{0-1-1}, \textit{1-0-0}, \textit{1-0-1}, \textit{1-1-0}, \textit{1-1-1}\}$$

The naming of outcomes implies our preferred social choice rule's outcome for each situation, assuming perfect honesty. For every type of profile, $\theta = (\theta_1, \theta_2, \theta_3)$, of the society, payoff corresponding to each outcome is given by the following matrix. Circled entries are our social choice rule, which maps each state to the efficient fair outcome.

	0 – 0 – 0	**0 – 0 – 1**	**0 – 1 – 0**	**0 – 1 – 1**	**1 – 0 – 0**	**1 – 0 – 1**	**1 – 1 – 0**	**1 – 1 – 1**
0, 0, 0	(1,1,1)	1, 1, 0	1, 0, 1	5/4, 0, 0	0, 1, 1	0, 5/4, 0	0, 0, 5/4	0, 0, 0
0, 0, 1	1, 1, 0	(1,1,1)	5/4, 0, 0	1, 0, 1	0, 5/4, 0	0, 1, 1	0, 0, 0	0, 0, 5/4
0, 1, 0	1, 0, 1	5/4, 0, 0	(1,1,1)	1, 1, 0	0, 0, 5/4	0, 0, 0	0, 1, 1	0, 5/4, 0
0, 1, 1	5/4, 0, 0	1, 0, 1	1, 1, 0	(1,1,1)	0, 0, 0	0, 0, 5/4	0, 5/4, 0	0, 1, 1
1, 0, 0	0, 1, 1	0, 5/4, 0	0, 0, 5/4	0, 0, 0	(1,1,1)	1, 1, 0	1, 0, 1	5/4, 0, 0
1, 0, 1	0, 5/4, 0	0, 1, 1	0, 0, 0	0, 0, 5/4	1, 1, 0	(1,1,1)	5/4, 0, 0	1, 0, 1
1, 1, 0	0, 0, 5/4	0, 0, 0	0, 1, 1	0, 5/4, 0	1, 0, 1	5/4, 0, 0	(1,1,1)	1, 1, 0
1, 1, 1	0, 0, 0	0, 0, 5/4	0, 5/4, 0	0, 1, 1	5/4, 0, 0	1, 0, 1	1, 1, 0	(1,1,1)

Table 4.1: Payoff matrix and social choice set for Example 4.1

For deception $\alpha(\theta) = \begin{cases} (1,1,1) & : \theta = (1,1,1) \\ (0,0,0) & : otherwise. \end{cases}$ and with $\epsilon = 0.75$, there is no $i \in N$ and $r : \Theta_{-i} \to A$ satisfying both conditions of epsilon-Bayesian Monotonicity for this payoff matrix and social choice rule.

4.2 EBM does not imply BM

This example is constructed upon a similar setup. Again, there are 3 agents, N={1,2,3} which can be type 0 or type 1, ($\Theta_i = \{0,1\}$). This is Epsilon Bayesian Monotonic for $\epsilon = 0.75$ but not Bayesian Monotonic.

For every type of profile, $\theta = (\theta_1, \theta_2, \theta_3)$, of the society, payoff corresponding to each outcome is given by the following matrix. Circled entries are our social choice rule, which maps each state to the efficient fair outcome.

For deception $\alpha'(\theta) = \begin{cases} (0,0,1) & : \theta = (0,1,0) \\ (0,0,0) & : \theta = (1,0,1) \\ \theta & : otherwise. \end{cases}$, there is no $i \in N$ and $r : \Theta_{-i} \to A$ satisfying both conditions of Bayesian Monotonicity for this payoff matrix and social choice rule.

	$\mathbf{0-0-0}$	$\mathbf{0-0-1}$	$\mathbf{0-1-0}$	$\mathbf{0-1-1}$	$\mathbf{1-0-0}$	$\mathbf{1-0-1}$	$\mathbf{1-1-0}$	$\mathbf{1-1-1}$
$\mathbf{0,0,0}$	(1,1,1)	$0,0,0$	$0,0,0$	$17/4,0,0$	$0,0,0$	$0,17/4,0$	$0,0,0$	$0,0,0$
$\mathbf{0,0,1}$	$0,0,0$	(1,1,1)	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$
$\mathbf{0,1,0}$	$0,0,0$	$0,0,0$	(1,1,1)	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$
$\mathbf{0,1,1}$	$0,0,0$	$0,0,0$	$0,0,0$	(1,1,1)	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$
$\mathbf{1,0,0}$	$17/4,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	(1,1,1)	$0,0,0$	$0,0,0$	$0,0,0$
$\mathbf{1,0,1}$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	(1,1,1)	$0,0,0$	$0,0,0$
$\mathbf{1,1,0}$	$0,0,17/4$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	(1,1,1)	$0,0,0$
$\mathbf{1,1,1}$	$0,0,0$	$0,0,17/4$	$0,17/4,0$	$0,0,0$	$0,0,0$	$0,0,0$	$0,0,0$	(1,1,1)

Table 4.2: Payoff matrix and social choice set for Example 4.2

CHAPTER 5

CONCLUSION

This thesis analyzes full implementation of a social choice rule via epsilon-Bayesian equilibrium and defines corresponding conditions, namely Epsilon bounded Incentive Compatibility (EIC) and Epsilon Bayesian Monotonicity (EBM). We prove that, together with Closure, these conditions are both necessary and sufficient for full implementation under our assumptions which includes the environment is epsilon-economic and there are at least three agents in the society.

Our analysis extends Jackson (1991) to consider bounded rationality, and gives examples in order to show that Bayesian implementable and epsilon-Bayesian implementable sets are not nested with each other. Our results show that full implemenation via Epsilon-Bayesian Equilibrium is possible when the conditions in Jackson (1991) are modified considering bounded rationality.

There are two main reasons that makes this valuable. First, as we exemplify, there are some social choice sets which are not Bayesian implementable as defined in Jackson (1991), but nevertheless is implementable via epsilon-Bayesian implementation. Second, when a social planner takes on a more behavioral approach, he may find epsilon-Bayesian equilibrium more sensible under bounded rationality hypothesis.

Possible extensions are analyzing incomplete information case with different levels of boundedness for each agent, or dealing with non-economic environments.

BIBLIOGRAPHY

Arrow, J., 1963. Social Choice and Individual Values. Yale University Press.

Barlo, M., Dalkiran, N., 2009. Epsilon-nash implementation. Economics Letters 102 (1), 36–38.

Barlo, M., Dalkiran, N., 2014. Epsilon ex-post implementation.

Benoît, J.-P., Ok, E. A., 2006. Maskin's theorem with limited veto power. Games and Economic Behavior 55 (2), 331–339.

Bergemann, D., Morris, S., 2008. Ex post implementation. Games and Economic Behavior 63 (2), 527–566.

Gibbard, A., 1973. Manipulation of voting schemes: a general result. Econometrica, 587–601.

Groves, T., Ledyard, J., 1977. Optimal allocation of public goods: A solution to the" free rider" problem. Econometrica, 783–809.

Hurwicz, L., 1979. Outcome functions yielding walrasian and lindahl allocations at nash equilibrium points. The Review of Economic Studies, 217–225.

Jackson, M. O., 1991. Bayesian implementation. Econometrica, 461–477.

Maskin, E., 1999. Nash equilibrium and welfare optimality. Review of Economic Studies 66, 23–38.

Palfrey, T. R., Srivastava, S., 1987. On bayesian implementable allocations. The Review of Economic Studies 54 (2), 193–208.

Palfrey, T. R., Srivastava, S., 1989. Implementation with incomplete information in exchange economies. Econometrica, 115–134.

Postlewaite, A., Schmeidler, D., 1986. Implementation in differential information economies. Journal of Economic Theory 39 (1), 14–33.

Radner, R., 1980. Collusive behavior in noncooperative epsilon-equilibria of oligopolies with long but finite lives. Journal of economic theory 22 (2), 136–154.

Satterthwaite, M., 1975. Strategy-proofness and arrow's conditions: Existence and correspondence theorems for voting procedures and social welfare functions. Journal of economic theory 10 (2), 187–217.

Schmeidler, D., 1980. Walrasian analysis via strategic outcome functions. Econometrica, 1585–1593.

APPENDIX

Now, we explain the C++ code we use for checking examples 4.1 and 4.2. Below are the definitions of processes and functions.

- *fstate* function finds corresponding states for each row in our examples.
- *ffpos* function finds all possible states for agent i, given his state.
- *funa* function finds the result of $f \circ \alpha(\theta)$ for given a social choice rule f, and a deception α.
- *runa* function does the similar for a reward function $r : \Theta_{-i} \to A$.
- *pref1* function checks non-strict preference and also considers degree of boundedness of rationality. Since we are only checking this for (2.1), format of *f1* and *f2* is defined in a way that is compatible to that condition.
- *pref2* function checks strict preference and also considers degree of boundedness of rationality. Since we are only checking this for (2.2), format of *f1* and *f2* is defined in a way that is compatible to that condition.
- *halfdec* function finds the deception profile, when agent i tells the truth. What we will find via this function is a mapping from possible states for agent i, to itself. More formally this gives us $\alpha_{-i}(\theta_i)$ as in (2.1).

- There are 8^8 possible deceptions. Because of memory constraints, all deceptions are defined within-loop. That is different for possible reward functions, since there are only $8^4 = 4096$ possible r. Those functions are stored in posg array, and called when needed.

- Remaining parts are straightforward, code checks whether there is $r : \Theta_{-i} \to A$, $i \in N$ as in Definition 7, which satisfies 2.1 and 2.2.

- Code asks which boundedness level should be used, and which example to analyze. After taking these inputs, program will print corresponding whistleblowers, types of agents and reward functions to each deceptions. If there is a problematic deception, that is with no possible reward function no matter what type of which player is chosen, the code will print out that deception and stop checking. If there is not, it will continue to check until counter hits 16777216 and, this will show that given payoff matrix is Bayesian monotonic, in the default or the epsilon bounded sense.

A The Code

Below, we include C++ code which is explained above.

```
#include <stdio.h>
#include <conio.h>
#include <vector>
#include <iostream>
#include <string.h>
using namespace std;

//states
//0-1-2-3-4-5-6-7
```

```
int Theta[8][3]=
{
{0,0,0},{0,0,1},{0,1,0},{0,1,1},{1,0,0},{1,0,1},{1,1,0},{1,1,1},


};
//4.01
float U1[3][8][8]=
       { {
           {1,1,1,5/4,0,0,0,0},
           {1,1,5/4,1,0,0,0,0},
           {1,5/4,1,1,0,0,0,0},
           {5/4,1,1,1,0,0,0,0},
           {0,0,0,0,1,1,1,5/4},
           {0,0,0,0,1,1,5/4,1},
           {0,0,0,0,1,5/4,1,1},
           {0,0,0,0,5/4,1,1,1}
           },
           {
           {1,1,0,0,1,5/4,0,0},
           {1,1,0,0,5/4,1,0,0},
           {0,0,1,1,0,0,1,5/4},
           {0,0,1,1,0,0,5/4,1},
           {1,5/4,0,0,1,1,0,0},
           {5/4,1,0,0,1,1,0,0},
           {0,0,1,5/4,0,0,1,1},
           {0,0,5/4,1,0,0,1,1}
           },
           {
           {1,0,1,0,1,0,5/4,0},
```

```
            {0,1,0,1,0,1,0,5/4},
            {1,0,1,0,5/4,0,1,0},
            {0,1,0,1,0,5/4,0,1},
            {1,0,5/4,0,1,0,1,0},
            {0,1,0,5/4,0,1,0,1},
            {5/4,0,1,0,1,0,1,0},
            {0,5/4,0,1,0,1,0,1}
            }, };
        //4.02
float U2[3][8][8]=
        { {
            {1,0,0,17/4,0,0,0,0},
            {0,1,0,0,0,0,0,0},
            {0,0,1,0,0,0,0,0},
            {0,0,0,1,0,0,0,0},
            {17/4,0,0,0,1,0,0,0},
            {0,0,0,0,0,1,0,0},
            {0,0,0,0,0,0,1,0},
            {0,0,0,0,0,0,0,1}
                        },
            {
            {1,0,0,0,0,17/4,0,0},
            {0,1,0,0,0,0,0,0},
            {0,0,1,0,0,0,0,0},
            {0,0,0,1,0,0,0,0},
            {0,0,0,0,1,0,0,0},
            {0,0,0,0,0,1,0,0},
            {0,0,0,0,0,0,1,0},
            {0,0,17/4,0,0,0,0,1}
```

```
          },
          {
          {1,0,0,0,0,0,0,0},
          {0,1,0,0,0,0,0,0},
          {0,0,1,0,0,0,0,0},
          {0,0,0,1,0,0,0,0},
          {0,0,0,0,1,0,0,0},
          {0,0,0,0,0,1,0,0},
          {17/4,0,0,0,0,0,1,0},
          {0,17/4,0,0,0,0,0,1}
          }, };
//FSTATE------finds the state
   ---------------------------------------
std::vector<int> fstate(int st)
{
int i;
std::vector<int> state;
state.resize(3);
 for ( i = 0; i < 3; i++)
  {
   state[i] = Theta[st][i];
 }
return(state);
}
//FPOS-possible states for calculating pref
   ----------------------
std::vector<int> ffpos(int tip,int pl)
{
int count=0;
```

```
int k;
std::vector<int> possible;
possible.resize(4);
 for ( k = 0; k < 8; k++)
  {
 std::vector<int> ol=fstate(k);
 if (tip==ol[pl])
 {
 possible[count]=k;
 count++;
 }
 }
return(possible);
}
//Funiona-foalpha
   ------------------------------------------------------
std::vector<int> funiona(std::vector<int> f,int dec[8][1])
{
int i;
std::vector<int> y;
y.resize(8);
 for ( i = 0; i < 8; i++)
  {
   y[i] = f[dec[i][0]];
 }
return(y);
}
//Guniona-goalpha
   ------------------------------------------------------
```

```
std::vector<int> guniona(std::vector<int> g,int dec[4][1])
{
int i;
std::vector<int> y;
y.resize(4);
 for ( i = 0; i < 4; i++)
  {
   y[i] = g[dec[i][0]];
 }
return(y);
}
//Pref1-----R-------------------------------------------------
bool pref1(std::vector<int> pos,std::vector<int> f1,std::vector
   <int> f2,float U[3][8][8],int pl,float eps)
{
       float out11=U[pl][pos[0]][f1[pos[0]]]+U[pl][pos[1]][f1[
          pos[1]]]+U[pl][pos[2]][f1[pos[2]]]+U[pl][pos[3]][f1[
          pos[3]]];
       float out21=U[pl][pos[0]][f2[0]]+U[pl][pos[1]][f2[1]]+U[
          pl][pos[2]][f2[2]]+U[pl][pos[3]][f2[3]];
       if (out11>=(out21-4*eps))
       {
       return(1);
       }
       else
       {
       return(0);
       }
}
```

```
//Pref2-----P-------------------------------------------------
bool pref2(std::vector<int> pos,std::vector<int> f1,std::vector
   <int> f2,float U[3][8][8],int pl,float eps)
{
       float out11=U[pl][pos[0]][f1[0]]+U[pl][pos[1]][f1[1]]+U[
          pl][pos[2]][f1[2]]+U[pl][pos[3]][f1[3]];
       float out21=U[pl][pos[0]][f2[pos[0]]]+U[pl][pos[1]][f2[
          pos[1]]]+U[pl][pos[2]][f2[pos[2]]]+U[pl][pos[3]][f2[
          pos[3]]];
       if (out11>(out21+4*eps))
       {
       return(1);
       }
       else
       {
       return(0);
       }
}

//halfdec-------------half deception, one agent is honest
   ----------------------------------------
std::vector<int> halfdec(int wb,int dec[8][1],std::vector<int>
   pos)
{

       int halfdec1[4][3];
       int i,a,k;
       for ( i = 0; i < 4; i++)
```

```
       {
          halfdec1[i][0]=fstate(dec[pos[i]][0])[0];
          halfdec1[i][1]=fstate(dec[pos[i]][0])[1];
          halfdec1[i][2]=fstate(dec[pos[i]][0])[2];
          halfdec1[i][wb]=fstate(pos[i])[wb];
       }


       std::vector<int> y,l;
       l.resize(4);
       y.resize(4);
       for ( i = 0; i < 4; i++)
       {
                     for ( a = 0; a < 8; a++)
                     {
                     if (halfdec1[i][0]==Theta[a][0])
                     {
                     if (halfdec1[i][1]==Theta[a][1])
                     {
                     if (halfdec1[i][2]==Theta[a][2])
                     {
l[i]=a;
break;
                     }
                     }
                  }
       }
}
for ( i = 0; i < 4; i++)
       {
```

```
for ( a = 0; a < 4; a++)
                        {
                if(l[i]==pos[a])
                {
                        y[i]=a;
                        break;
                }
}
}
        return(y);
}
//MAIN FUNCTION
int main(void)
{
            float eps;
            int a1,a2,a3;
            int aa;
            float U[3][8][8];
cout << "Please␣choose␣epsilon-␣degree␣of␣bounded␣rationality:␣
   ";
    cin >> eps;
cout <<"Please␣choose␣the␣example␣payoff␣matrix␣for␣working␣on.
   ␣\n␣1␣for␣4.1,␣2␣for␣4.2␣:";
cin>> aa;
switch(aa)
{
case 1:
for(a1=0;a1<3;a1++)
{
```

```
        for(a2=0;a2<8;a2++)
        {
                for(a3=0;a3<8;a3++)
                {
                        U[a1][a2][a3]=U1[a1][a2][a3];
                }
        }
}
break;
case 2:for(a1=0;a1<3;a1++)
{
        for(a2=0;a2<8;a2++)
        {
                for(a3=0;a3<8;a3++)
                {
                        U[a1][a2][a3]=U1[a1][a2][a3];
                }
        }
}
break;
}
printf(":::Working:::\n\n");
//Definition of F
int i1,i2,i3,i4,i5,i6,i7,i8;
int ff[8][1]={0,1,2,3,4,5,6,7};
//int ff[8][1]={0,0,0,0,0,0,0};
std::vector<int> f;
f.resize(8);
f[0]=ff[0][0];
```

```
f[1]=ff[1][0];
f[2]=ff[2][0];
f[3]=ff[3][0];
f[4]=ff[4][0];
f[5]=ff[5][0];
f[6]=ff[6][0];
f[7]=ff[7][0];
int say=1;
int pog[4][8]=
{
       {0,1,2,3,4,5,6,7},
       {0,1,2,3,4,5,6,7},
       {0,1,2,3,4,5,6,7},
       {0,1,2,3,4,5,6,7}
};
int posg[4][4097];
               for(i1=0;i1<8;i1++)
{
               for(i2=0;i2<8;i2++)
{
               for(i3=0;i3<8;i3++)
{
               for(i4=0;i4<8;i4++)
{
posg[0][say]=pog[0][i1];
posg[1][say]=pog[1][i2];
posg[2][say]=pog[2][i3];
posg[3][say]=pog[3][i4];
say++;
```

```
}
}
}
}
printf("Deception␣check␣starts\n\n");
int d1,d2,d3,d4,d5,d6,d7,d8,wb,sg,ii,iic,k,pl,count,deccount;
deccount=1;
bool p,r;
int wbb[4][1]={0,0,1,2};
int iii[3][1]={0,0,1};
                          for(d1=0;d1<8;d1++)
{
               for(d2=0;d2<8;d2++)
{
               for(d3=0;d3<8;d3++)
{
               for(d4=0;d4<8;d4++)
{
               for(d5=0;d5<8;d5++)
{
               for(d6=0;d6<8;d6++)
{
               for(d7=0;d7<8;d7++)
{
               for(d8=0;d8<8;d8++)
{
int dec[8][1]={d1,d2,d3,d4,d5,d6,d7,d8};
std::vector<int> funa;
funa=funiona(f,dec);
```

```
count=0;
printf("%d␣",deccount);
deccount++;
for(pl=0;pl<4;pl++)
{
wb=wbb[pl][0];
for(sg=0;sg<4097;sg++)
{
std::vector<int> g;
g.resize(4);
g[0]=posg[0][sg];
g[1]=posg[1][sg];
g[2]=posg[2][sg];
g[3]=posg[3][sg];
if((pref1(ffpos(0,wb),f,g,U,wb,eps)==1)&&(pref1(ffpos(1,wb),f,g
   ,U,wb,eps)==1))
{
for(iic=0;iic<3;iic++)
{
ii=iii[iic][0];
std::vector<int> pos;
pos=ffpos(ii,wb);
std::vector<int> halfdecc;
halfdecc=halfdec(wb,dec,pos);
int halfdeccc[4][1];
halfdeccc[0][0]=halfdecc[0];
halfdeccc[1][0]=halfdecc[1];
halfdeccc[2][0]=halfdecc[2];
halfdeccc[3][0]=halfdecc[3];
```

```
std::vector<int> gunhalfdec;
gunhalfdec=guniona(g,halfdeccc);
p=pref2(pos,gunhalfdec,funa,U,wb,eps);
        if(p==1)
        {
        count++;
        }
        if (count>0)
{
break;
}
}
}
if (count>0)
{
        printf("-deception:␣%d␣",dec[0][0]+1);
        printf("%d␣",dec[1][0]+1);
    printf("%d␣",dec[2][0]+1);
        printf("%d␣",dec[3][0]+1);
        printf("%d␣",dec[4][0]+1);
        printf("%d␣",dec[5][0]+1);
        printf("%d␣",dec[6][0]+1);
        printf("%d␣",dec[7][0]+1);
        printf("agent:␣%d␣",wb+1);
        printf("type:␣%d␣",ii);
        printf("reward:␣%d",g[0]+1);
    printf("%d",g[1]+1);
        printf("%d",g[2]+1);
        printf("%d\n",g[3]+1);
```

```
        posg[0][0]=g[0];
        posg[1][0]=g[1];
        posg[2][0]=g[2];
        posg[3][0]=g[3];
        wbb[0][0]=wb;
        iii[0][0]=ii;
break;
}
}
if (count>0)
{
break;
}
}
if (count==0)
{
        if((d1==0)&&(d2==1)&&(d3==2)&&(d4==3)&&(d5==4)&&(d6==5)
           &&(d7==6)&&(d8==7))
        {
    count=1;
        }
}

if (count==0)
{
        printf("Problematic␣deception:␣%d␣",d1+1);
        printf("%d␣",d2+1);
        printf("%d␣",d3+1);
        printf("%d␣",d4+1);
```

```
        printf("%d␣",d5+1);
        printf("%d␣",d6+1);
        printf("%d␣",d7+1);
        printf("%d\n",d8+1);
break;
}
}

                if (count==0)
{
break;
}
}
                if (count==0)
{
break;
}
                }
                 if (count==0)
{
break;
}
                }
                if (count==0)
{
break;
}
                }
                if (count==0)
```

```
{
break;
}
                }
                if (count==0)
{
break;
}
                }
                if (count==0)
{
break;
}
                }


  getch();
}
```