Strategic information transmission: a mathematica tool for analysis

Dickhaut, John and Kaplan, Todd R and Mukherji, Arijit

Universities of Exeter and Haifa

September 1992

Strategic Information Transmission
A *Mathematica* Tool for Analysis

John Dickhaut,
Todd Kaplan and
Arijit Mukherji

General Introduction

Economists and other applied researchers use game theory to study industrial organization, financial markets, and the theory of the firm. In an earlier article in the *Mathematica* Journal, [Dickhaut and Kaplan 1991] present a procedure for solving two-person games of complete information. In many applications, however, "asymmetric information" is a central issue. By asymmetric information, we mean that one party has access to information that the other party lacks. The branch of game theory that deals with this problem is known as "games of incomplete information"; the formal model is discussed in [Harsanyi 1967]. [Myerson 1991, Tirole 1989] et al, discuss the applications but do not focus on computational procedures. We provide, in this article, an application of *Mathematica* to games of incomplete information that should be of interest for two reasons: (i) as a basis for thinking about solutions to games of incomplete information, and (ii) as an approach to understanding the particular application presented here, namely, the effect of strategic information transmission in firms and markets.

A game of strategic information transmission is played by a Sender and a Receiver. The Sender has private information about some random state of nature, which he will have the opportunity to communicate to the Receiver. The Receiver can take an action that affects the welfare of both parties. The Sender and Receiver have preferences (defined in terms of utility functions) over states **m** and actions **y**. Since the state affects the payoffs, players may wish to condition actions on the state. However, the Receiver and Sender can have divergent preferences, with a parameter **b** measuring the extent to which they disagree. Thus, the Sender's favorite action for a given state, m, is potentially different from the Receiver's favorite action for the same state. Once Nature has chosen a particular state, the Sender may decide not to reveal fully his information about the state, for such a policy could guarantee that the Receiver's favorite action will be selected. However, even though there is some divergence in preferences, there may still be an underlying gain that can be achieved from coordination. Since the Sender does, in general, send a message that is partially informative, the Receiver does learn something about **m** given the message, as he takes account of the incentives of the Sender. Given this partial knowledge, the Receiver chooses an optimal action **y** to maximize his own payoff. The Sender decides on the degree of disclosure after considering how the Receiver will respond to the message sent.

The strategic element in this setting is the fact that the Sender considers the effect of the disclosure on the Receiver's action choice, and the Receiver considers the impact of how he responds to information on the Sender's disclosure plans. Combined appropriately with statistical conditions that reflect the Receiver's beliefs, the main solution concept used in games of incomplete information is that of Bayesian-Nash equilibrium. Broadly speaking, this states that the Sender's

disclosure strategy must be optimal given the Receiver's action strategy, and vice versa.

The following examples illustrate the forces outlined above. First, consider a lobbyist who has private information about the efficacy of a desired program. The lobbyist may realize that disclosing some of the information could influence legislators to support the program; however, disclosing too much might reveal some defects which could reduce the appropriation. Similarly, a sales representative could truthfully report potential sales so that the company would be well-served, but leaving himself some slack could protect him from unforeseen demand shifts. In either case the Receiver of the information will no doubt be aware of the Sender's motives. Given each party can properly anticipate the other's intentions, the crucial question is what messages are transmitted. Crawford and Sobel show that as long as interests are not perfectly aligned, information is conveyed noisily (that is, private information is not fully revealed).

The package **strategic** solves two-person Sender-Receiver games in which the Sender holds private information about the true state, and the welfare of the parties is state-dependent. In the above examples the appropriation by Congress and the production budget chosen by the supervisor, respectively, affect the welfare of the lobbyist and the sales representative. The inputs to the program are the tastes (utility functions) of the parties, and the underlying state uncertainty. For the case where private information can assume finitely many values, the output of the program includes the Sender's message(depending on which state he observes), the action that the Receiver will take in response to each potential message, and the Receiver's inferences. If there are multiple equilibria, they are distinguished according to the fineness of the disclosure. Such equilibria can be ranked.

The package solves the problem both when the states are finite, and when they are an interval. For finitely many states, the program recursively identifies the set of partitions that constitute the potential messages of the Sender; it uncovers fixed points from best responses using pure functions. With a continuum, the package solves difference equations (numerically), then graphs the key features of partitions, as well as the change in welfare .

Example 1: A Simple Numerical Illustration

We consider a world in which there are two equally likely states of nature, s1 and s2. There are two players, a Sender and Receiver. The Sender alone observes the state and delivers a message to the Receiver. The Receiver takes one of two actions, a1 or a2. Payoffs to both parties depend on the coordination of state and action. When preferences are aligned, both Sender and Receiver prefer the same action be taken; this gives the Sender has an incentive to report truthfully. When preferences are not aligned, naive truthful reporting would be foolhardy permitting the reciever to implement his favorite action to the Sender's detriment. A setting in which preferences are aligned is

		Actions of the Receiver	
		a1	a2
States	s1	(1, 4)	(0, 0)
	s2	(0, 0)	(1, 1)

The first number of a cell is the payoff to the Sender and the second the payoff to the Receiver.

A setting in which preferences are not aligned is

		Actions of the Receiver	
		a 1	a 2
States	s 1	(1, 0)	(0, 4)
	s2	(0, 1)	(1, 0)

The messages that can be sent are nonempty subsets of {s1,s2}. An equilibrium consists of three components.

a)a strategy for the Sender mapping {s1,s2} to messages,
b)a mapping from messages to actions for the Receiver, and
c)a set of posterior beliefs of the Receiver given the message.

For an equilibrium in such a game of incomplete information, the three components must be self-fulfilling in the sense that

1)The strategy in (a) has to be optimal given (b) and (c).
2)The strategy in (b) has to be optimal given (a) and (c).
3)The beliefs in (c) must be derived from Bayes rule given (a).

When preferences are not aligned, the equilibrium involves the Sender sending the message {s1,s2} regardless of the state. The Receiver's posteriors are the same as his priors, so he chooses action a2. When preferences are aligned there are two equilibria, one of which is preferred by both players. In the first equlibirum, the Sender truthfully reports the state, the Receiver believes the Sender, and takes the appropriate action. This equilibrium is preferred by both Sender and Receiver to the equilibrium in which the Sender's message is {s1,s2}, beliefs of the Receiver are unchanged, and the Reciever takes action a1.

Example 2: A More Complex Illustration

The structure of the problem can be extended to more complicated settings. The way that we characterize the conflict of interest is in terms of the distance of preferred actions given a state. In the examples below we assume that each state is equally likely. First, we consider an extended structure of the problem when the preferences are aligned. In this case, given a particular action, both the Sender and Receiver would prefer that the Reciever take the action which is reflected in the diagonal payoffs.

		Actions of the Receiver			
		a 1	a 2	a 3	a 4
	s 1	(242, 274)	(242, 214)	(176, 145)	(105, 73)
State	s 2	(176, 214)	(242, 274)	(242, 214)	(176, 145)
	s 3	(105, 145)	(176, 214)	(242, 274)	(242, 214)
	s 4	(31, 73)	(105, 145)	(176, 214)	(242, 242)

By contrast, the following is a case in which preferences are not aligned:

		Actions of the Receiver			
		a 1	a 2	a 3	a 4
State	s1	(141, 274)	(210, 214)	(270, 145)	(210, 73)
	s2	(69, 214)	(141, 274)	(210, 214)	(270, 145)
	s3	(-6, 145)	(69, 214)	(141, 274)	(210, 214)
	s4	(-83, 73)	(-6, 145)	(69, 214)	(141, 242)

In each of these cases the set of messages would be subsets of {s1,s2,s3,s4}. However the equilibria that would emerge would be quite different. For the setting in which preferences are not aligned, the equilbrium signal would be only the set{s1,s2,s3,s4}, the Receiver's beliefs would be the same as prior to receiving the message, and the action taken by the Receiver would consistently be a2, or a3.

In the setting where preferences are aligned, there are several different equilibria. There is the completely informative equilibrium, in which the Sender reveals the state truthfully, the Receiver assigns probability 1 to the state reported and 0 to all other states, and the Receiver takes the his preferred action(which the sender also prefers). There are, however, several other equilbria which involve different messages conveyed by the Sender. In fact the following partitions of {s1,s2,s3,s4} constitute equilibria in the setting when preferences are aligned.

{{{s1}, {s2}, {s3}, {s4}},{{s1}, {s2, s3, s4}}, {{s1},{s1,s2,s3,s4}},
{{s1, s2}, {s3,s4}},{{s1,s2},{s2,s3,s4}}
{{{s1}, {s2}, {s3, s4}},{{s1},{s2},{s2,s3,s4}},{{s1},{s2,s3},{s4}},
{{s1}, {s2, s3}, {s4}}
{{s1,s2},{s3},{s3,s4}}, {{s1, s2}, {s3},{s4}}, {{s1},{s2},{s3},{s4}},{{s1},{s2},{s3},{s3,s4}}}

The notation {{s1},{s1,s2,s3,s4}} means that the sender is indifferent between sending{s1} or{s1,s2,s3,s4} when s1 occurs.

There are intermediate settings, i.e., settings in which the partitions reveal some information but are not fully truth-revealing. For example:

		Actions of the Receiver			
		a1	a2	a3	a4
State	s1	(216, 274)	(276, 214)	(216, 145)	(147, 73)
	s2	(147, 214)	(216, 274)	(276, 214)	(216, 145)
	s3	(75, 145)	(147, 214)	(216, 274)	(276, 214)
	s4	(0, 73)	(75, 145)	(147, 214)	(216, 242)

The equilibria for this game are supported by the following partitions,
{{{s1}, {s2, s3, s4}},{{s1,s2,s3,s4}}} but not {{s1}, {s2}, {s3}, {s4}}.

Finding Solutions to the Finite Problem

■ The Three Functions: EqMessages, EqBeliefs, and ReActions

The three functions **EqMessages,EqBeliefs**, and **ReActions** are used to solve for equilibria of Sender-Receiver games with a finite number of states and actions. The function **EqMessages** assumes that both states and actions can be ordered as sequences of consecutive integers starting with one. **EqMessages** assumes that utilities for Sender and Receiver (defined on the Cartesian Product of states and actions) and state probabilities are known. The output from this function consists of sets of subsets of integers, corresponding to partitions of the set of states that can be parts of equilibria in the specified Sender-Receiver game. In the following section, we will introduce the functions and show how they are use to solve problems.

■ Applications

We begin by defining the utility functions for Receiver and Sender and a probability function. At this point we load in the package, **strategic**, which contains functions that will be used in this notebook.

```
SetDirectory["/ari"];

<<strategic

Ur[y_,m_]:=-(y-m)^2;

Us[y_,m_,b_]:=- (y-m-b)^2;
Prob[x_]:=1/4;
```

To see what these imply consider the following

```
ReceiversUtilities[A_,  S_,Ur_] :=
  Table[Ur[a,s],
  {a, 1, Length[A]},
  {s, 1, Length[S]}]
```

In the case when the actions and states are the sets {1,2,3,4}, we get the entries for the Receiver's payoffs illustrated in Example #2.

```
MatrixForm[ReceiversUtilities[Range[4],Range[4],Ur]]
```

```
0    −1   −4   −9
−1   0    −1   −4
−4   −1   0    −1
−9   −4   −1   0
```

For a specific state, we represent the degree of divergence of utilities between Sender and Receiver by an index paramater b.

```
SendersPossibleUtilities[A_, s_, b_] :=
  Table[Us[a,s,b],
  {a, 1, Length[A]}]
```

For the case in which preferences are aligned, we take b to be .5. The Sender's utilities look like the following:

```
b=.5;
MatrixForm[
Table[SendersPossibleUtilities[Range[4],s,b],{s,1,4}]]
```

```
-0.25    -0.25    -2.25    -6.25
-2.25    -0.25    -0.25    -2.25
-6.25    -2.25    -0.25    -0.25
-12.25   -6.25    -2.25    -0.25
```

Now to solve this problem we use the function **EqMessages**, which is defined by

```
EqMessages[states_, actions_,b_,Ur_,Us_,Prob_] :=
  Join[{{Range[states]}}, Complement[Flatten[
    Table[
         Table[IsEquilibrium[Range[states],
         Range[actions],Flatten[
             completegroup /@
             partition[Range[states],i], 1][[j]],
             b,Ur,Us,Prob],
          {j, 1, Length[Flatten[completegroup /@
           partition[Range[states], i], 1]]}],
      {i, 2, states}], 1],
   {{}}]]
```

With aligned preferences, we write the following:

```
b=.5;
Example2=EqMessages[4,4,.5,Ur,Us,Prob]
```

```
{{{1, 2, 3, 4}}, {{1}, {2, 3, 4}}, {{1}, {1, 2, 3, 4}},
  {{1, 2}, {3, 4}}, {{1, 2}, {2, 3, 4}}, {{1}, {2}, {3, 4}},
  {{1}, {2}, {2, 3, 4}}, {{1}, {1, 2}, {3, 4}},
  {{1}, {1, 2}, {2, 3, 4}}, {{1}, {2, 3}, {4}}, {{1}, {2, 3}, {3, 4}},
  {{1, 2}, {3}, {4}}, {{1, 2}, {3}, {3, 4}}, {{1, 2}, {2, 3}, {4}},
  {{1, 2}, {2, 3}, {3, 4}}, {{1}, {2}, {3}, {4}},
  {{1}, {2}, {3}, {3, 4}}, {{1}, {2}, {2, 3}, {4}},
  {{1}, {2}, {2, 3}, {3, 4}}, {{1}, {1, 2}, {3}, {4}},
  {{1}, {1, 2}, {3}, {3, 4}}, {{1}, {1, 2}, {2, 3}, {4}},
  {{1}, {1, 2}, {2, 3}, {3, 4}}}
```

There are many equilibria in this case. We can determine the beliefs associated with these equilibria in messages using the **EqBeliefs** function. That is, **EqBeliefs** shows the Receiver's posterior beliefs, given a message from the Sender when the Sender is playing an equilibrium strategy.

EqBeliefs[Example2,Prob]

```
{{{1/4, 1/4, 1/4, 1/4}}, {{1, 0, 0, 0}, {0, 1/3, 1/3, 1/3}},
  {{1, 0, 0, 0}, {1/4, 1/4, 1/4, 1/4}}, {{1/2, 1/2, 0, 0}, {0, 0, 1/2, 1/2}},
  {{1/2, 1/2, 0, 0}, {0, 1/3, 1/3, 1/3}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 1/3, 1/3, 1/3}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 1/3, 1/3, 1/3}},
  {{1, 0, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 0, 1}},
  {{1, 0, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 1/2, 1/2}},
  {{1/2, 1/2, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
  {{1/2, 1/2, 0, 0}, {0, 0, 1, 0}, {0, 0, 1/2, 1/2}},
  {{1/2, 1/2, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 0, 1}},
  {{1/2, 1/2, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 0, 1, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 0, 1}},
  {{1, 0, 0, 0}, {0, 1, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 0, 1, 0}, {0, 0, 0, 1}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 0, 1, 0}, {0, 0, 1/2, 1/2}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 0, 1}},
  {{1, 0, 0, 0}, {1/2, 1/2, 0, 0}, {0, 1/2, 1/2, 0}, {0, 0, 1/2, 1/2}}}
```

The actions taken in each setting are given by **ReActions**:

```
Table[ReActions[Range[4],Range[4],Example2[[j]],Ur,Prob],
    {j,1,Length[Example2]}]
```

```
{{{2, 3}}, {{1}, {3}}, {{1}, {2, 3}}, {{1, 2}, {3, 4}}, {{1, 2}, {3}},
  {{1}, {2}, {3, 4}}, {{1}, {2}, {3}}, {{1}, {1, 2}, {3, 4}},
  {{1}, {1, 2}, {3}}, {{1}, {2, 3}, {4}}, {{1}, {2, 3}, {3, 4}},
  {{1, 2}, {3}, {4}}, {{1, 2}, {3}, {3, 4}}, {{1, 2}, {2, 3}, {4}},
  {{1, 2}, {2, 3}, {3, 4}}, {{1}, {2}, {3}, {4}},
  {{1}, {2}, {3}, {3, 4}}, {{1}, {2}, {2, 3}, {4}},
  {{1}, {2}, {2, 3}, {3, 4}}, {{1}, {1, 2}, {3}, {4}},
  {{1}, {1, 2}, {3}, {3, 4}}, {{1}, {1, 2}, {2, 3}, {4}},
  {{1}, {1, 2}, {2, 3}, {3, 4}}}
```

For the case in which preferences are not aligned, we take b to be 2.0. The Senders utilities look like the following.

```
b=2.0;MatrixForm[
Table[SendersPossibleUtilities[Range[4],s,b],{s,1,4}]]
```

```
-4.    -1.    0.     -1.
-9.    -4.    -1.    0.
-16.   -9.    -4.    -1.
-25.   -16.   -9.    -4.
```

```
EqMessages[4,4,2.0,Ur,Us,Prob]
```

```
{{{1, 2, 3, 4}}}
```

We now turn to the setting in which the states and actions can fall on a continuum.

The Continuous Crawford-Sobel Model

■ Connection to the previous section.

So far we have discussed how to solve games of strategic infromation transmission where the set of states and of actions is finite. A probability mass function describes the relative likelihood of finitely many states. In many applications the set of states is modelled as a continuum. We assume that the state is a random variable with a known density function.

The model with a continuum of states (**m** in **M**) has the feature that the message space of the Sender consists of the set of all non-empty subsets of M. In equilibrium, if preferences are well-behaved (Utility functions increase in the action, are concave in the action, and marginal utility for the action changes monotonically with the state) then Crawford and Sobel (Econometrica 1982) show that

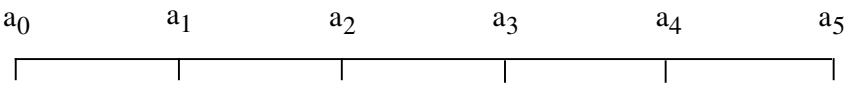
(i) Equilibria are partition equilibria.

(ii)Actions are a step function.

(iii)Multiple equilibria may exist.

Essentially the Sender's types (the states) can be divided into n different non-overlapping sub-intervals of **M**: over a particular region, all Sender types send the same message, inducing the Receiver to take the same action. Solving for a

partition involves finding the n-1 points where the sub-intervals meet. At these points, the Sender must be indifferent between claiming to be in one of the adjacent sub-intervals.



A five element partition.

A partition equilibrium is characterized by the solution to a system of difference equations.

In the first example, we replicate a simple illustration in Crawford and Sobel's original paper; in the second, we examine a variation on their model to address a question on which the literature has been silent.

■ Example 3: Crawford-Sobel's model (1982) of coordination with similar interests

■ Preliminaries

In this game, Nature chooses an **m** . The Sender is aware of the value of **m**, while the Receiver knows only the distribution function of **m**, which is uniform on the interval [0,1]. The Sender chooses a message (a subset of [0,1]) to send to the Receiver which may include information about **m**. Using this message, the Receiver chooses an action y so that he maximizes the expected value of his utility function.

We define an **equilibrium** to be a pair (m*(m), y*(m*(m)) where m* and y* can be correspondences, such that m*( ) is the argument that maximizes Us[y*(m(m)),m,b] over m( ) and y*( ) is the argument that maximizes E[Ur[y(m*(m)),m] | m ~ U[0,1],m*(m)] over y( ).

■ Existence of Fully Revealing Equilibria

There is no fully revealing equilibrium when b - 0 since in this case preferences are imperfectly aligned. One can see this by assuming that the Sender reveals his information **m** to the Receiver. In this case, the Receiver chooses y=**m**. Now the Sender's utility is

```
Clear[b];

Us[m,m,b]
```

$$-b^2$$

If the Sender, instead lies about **m**, then he could get a higher utility:

```
foc=D[Us[ym,m,b],ym]==0;
ans=Solve[foc,ym]
Us[ym,m,b]/.ans
```

```
{{ym -> b + m}}
```

```
{0}
```

In other words, the Sender wishes the Receiver to choose **y=b+m**, giving himself a utility of 0 rather than $-\mathbf{b}^2$. Knowing this the Receiver will never choose **y=m**. Therefore, the only candidates for types of equilibria are partial-disclosure equilibria or no-disclosure equilibria.

■ No disclosure equilibria

In the no-disclosure equilibrium, the Sender's message is completely uninformative: that is, he announces the entire set [0,1], so the Receiver learns nothing new. Now the Receiver chooses an action that maximizes his expected utility, the expectation being taken with respect to the prior distribution.

```
ExpUtil=Integrate[Ur[y,m],{m,0,1}]
```

$$-\left(\frac{1}{3}\right) + y - y^2$$

```
foc=D[ExpUtil,y]==0
```

```
1 - 2 y == 0
```

```
1 - 2 y == 0
```

```
1 - 2 y == 0
```

```
Solve[foc,y]
```

$$\{\{y \to \frac{1}{2}\}\}$$

$$\{\{y \to \frac{1}{2}\}\}$$

$$\{\{y \to \frac{1}{2}\}\}$$

The Sender has no incentive to change his no-disclosure strategy because the Receiver will still choose y=1/2, believing that any value of m in [0,1] is equally likely. The expected utilities in this equilibrium are, respectively for the Sender and the Receiver:

```
Simplify[Integrate[Us[1/2,m,b],{m,0,1}]]
Integrate[Ur[1/2,m],{m,0,1}]
```

$$-\left(\frac{1}{12}\right) - b^2$$

$$-\left(\frac{1}{12}\right)$$

■ Partition Equilibria

In a partition equilibrium, the set [0,1] is divided into partitions {a[0],...,a[n]} where a[0]=0, a[n]=1 and a[i+1] > a[i]. Each Sender announces the partition (a[i], a[i+1]). This could be interpreted as the Sender randomly choosing a point from the subinterval (a[i], a[i+1]) and the Receiver interpreting it as if the Sender's type were in ([a[i], a[i+1]).
Therefore, for each message in (a[i],a[i+1]), the Receiver chooses y so as to maximize his expected utility given posterior beliefs concentrated on the sub-interval (a[i], a[i+1]). We first compute **ExpUtil** which is the expected utility of the Receiver from choosing action y given that **m** is in the interval (a[i], a[i+1]). The **foc** is the first order condition with respect to y, and **ans** is the solution.

```
ExpUtil=Integrate[Ur[y,m],{m,a[i],a[i+1]}];
foc=D[ExpUtil,y]==0;
ans=Simplify[Solve[foc,y]]
```

$$\{\{y \to \frac{a[i] + a[1 + i]}{2}\}\}$$

We need to create a set of n different actions, one for each sub-interval in the partition. We define y[i] as the action taken if a message is sent in the partition (a[i],a[i+1]).

```
y[i_]:=Evaluate[(y/.ans)]
```

The Sender must have no incentive to lie, so

```
Us[y[i],m,b]>=Us[y[j],m,b]
```

$$\{-(-b - m + \frac{a[i] + a[1 + i]}{2})^2\} >= \{-(-b - m + \frac{a[j] + a[1 + j]}{2})^2\}$$

Notice the RHS is decreasing in j>i if b is small enough. For small enough b, the RHS is increasing in j<i as well Notice that adjacent partitions have a single point in common. At that point, both constraints must hold:

```
(Us[y[i+1],m,b]>=Us[y[i],m,b] &&
Us[y[i],m,b]>=Us[y[i+1],m,b] )/. {m->a[i+1]}
```

This is equivalent to having the Sender indifferent to being in either partition at the point in common.

```
Simplify[Us[y[i],m,b]==Us[y[i+1],m,b] /. {m->a[i+1]}]
```

$$\{\frac{-(2\ b - a[i] + a[1 + i])^2}{4}\} == \{\frac{-(-2\ b - a[1 + i] + a[2 + i])^2}{4}\}$$

Crawford and Sobel term these "indifference" or "arbitrage" conditions. We now need to check that the Receiver is taking an appropriate action given that he infers from a message that a Sender is in the interval (y[i],[y[i+1]):

```
y[4]
```

$$\left\{\frac{a[4] + a[5]}{2}\right\}$$

Notice that y is strictly increasing in i. In order for the Sender to have no incentive to change message partitions, his utility must be continuous in message sent. If this is not the case, **Us[y[i],m,b]>Us[y[i+1],m,b].** Notice that the y's are a step function.

■ Using the PartitionEq function:

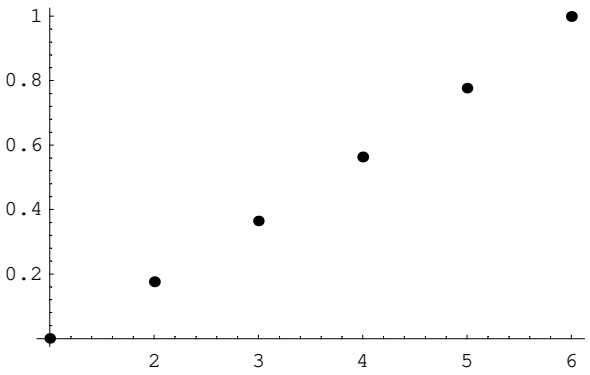
We now present the **PartitionEq** function. We first set up a group of equations **eq** which describe the indifference conditions. If there is a different utility function with no default equations provided, we run **Boundcond** which calculates the indifference conditions for that utility function. We append the boundary conditions to have a set of equations **eqns** to be solved for n+1 variables a[i], i=0,1,...n. The solutions are stored in g and tabulated in **ans1**. We prune the output to rule out complex solutions and also to guarantee that the numbers a[i] form a strictly increasing sequence. The output is a list of lists of indifference points, which completely characterizes the partition equilibrium. Note that if this list of lists is properly pruned, then it should be of dimensions 1 by n.

```
PartitionEq[n_,bb_,us_:1,eqopt_:1,vars_:1]:=Block[
{ans1,ans2,ans,eq,f,eqns,i,j,g},
If[n==1,Return[{{0.,1.}}] ];
If[eqopt===1,
  If[us===1,
    eq[i_]:=a[i+1]==2 a[i]-a[i-1]+4 bb,
    eq[i_]:=Evaluate[boundcond[us]/.b->bb];
  ];,
 If[vars===1 || Length[vars]!=3 ,eq[i_]:=Evaluate[eqopt/.b->bb] ,
  eq[i_]:=Evaluate[eqopt/.vars[[2]]->bb/.vars[[1]]->a/.vars[[3]]->i];];
];
eqns=Join[Table[eq[i],{i,1,n-1}],{a[0]==0,a[n]==1}];
g=NSolve[eqns,Table[a[i],{i,0,n}]];
ans1=Table[Table[a[i],{i,0,n}]/.g[[j]],
    {j,1,Length[g]}];
f[a_]:=Apply[And,Table[a[[i]]>a[[i-1]],{i,2,Length[a]}]];
ans=Select[ans1,f];
ans2=Select[ans1,Not[Apply[And,Map[NumberQ,#1]]] &];
If[Length[ans2]>0,ans=Join[ans,
{"Warning! Some partitions might not be increasing."},
ans2] ];
Return[ans];
]

boundcond[us_]:=Block[{y,m,a},
  ExpUtil=Integrate[us[y,m,0],{m,a[i],a[i+1]}];
  foc=D[ExpUtil,y]==0;
  y[i_]:=Evaluate[y/.Flatten[Simplify[Solve[foc,y]]]];
 Return[ us[y[i],a[i],b]==us[y[i-1],a[i],b]]
  ]
```

As a test, we work out what happens when we look for a 5 element partition with the preference parameter of 0.003.

```
atable=Flatten[PartitionEq[5,.003,1,1]];
ListPlot[atable, PlotStyle->{PointSize[.02]}]
```




```
-Graphics-
```

■ Welfare Comparisons

One can compute expected utilities of both parties by integrating each utility function over each partition element and then taking the **Sum** over different sub-intervals. The results are stored in **welfares** and **welfarer** for the Sender and Receiver, respectively.

```
ExpUtilSend[a_,b_]:=Sum[Integrate
[Us[(a[[i]]+a[[i+1]])/2,
m,b],{m,a[[i]],a[[i+1]]}],{i,1,Length[a]-1}];
ExpUtilRec[a_,b_]:=Sum[Integrate
[Ur[(a[[i]]+a[[i+1]])/2,m],
{m,a[[i]],a[[i+1]]}],{i,1,Length[a]-1}];
welfares=Table[ExpUtilSend
    [PartitionEq[n,.02],.02],{n,1,5}];
welfarer=Table[ExpUtilRec
    [PartitionEq[n,.02],.02],{n,1,5}];
```

Our next series of graphs helps to show how expected utility varies with the state, as the fineness of the partition changes. Crawford and Sobel show that although multiple partition equilibria can exist, they can be Pareto ordered (from best to worst) and ranked unanimously by both parties.

■ Comparative Statics

In this section we show how the utility changes as a function of the state given a partition, and then we determine how this relationship changes as the size of the partition changes.

```
parts=N[Table[PartitionEq[n,.05][[1]],{n,1,3}]];
toleft[a_,m_]:=Position[a,Join[Select
    [a,Function[x,x>m]],{1.}][[1]]][[1]]-1;
y[a_,m_]:=Block[{lpos},
    lpos=toleft[a,m];
    Return[(a[[lpos]]+a[[lpos+1]])/2] ];
graphs=Table[Plot[Us[y[parts[[n]],m],m,.05],
   {m,0,1},DisplayFunction->Identity],{n,1,3}]
```

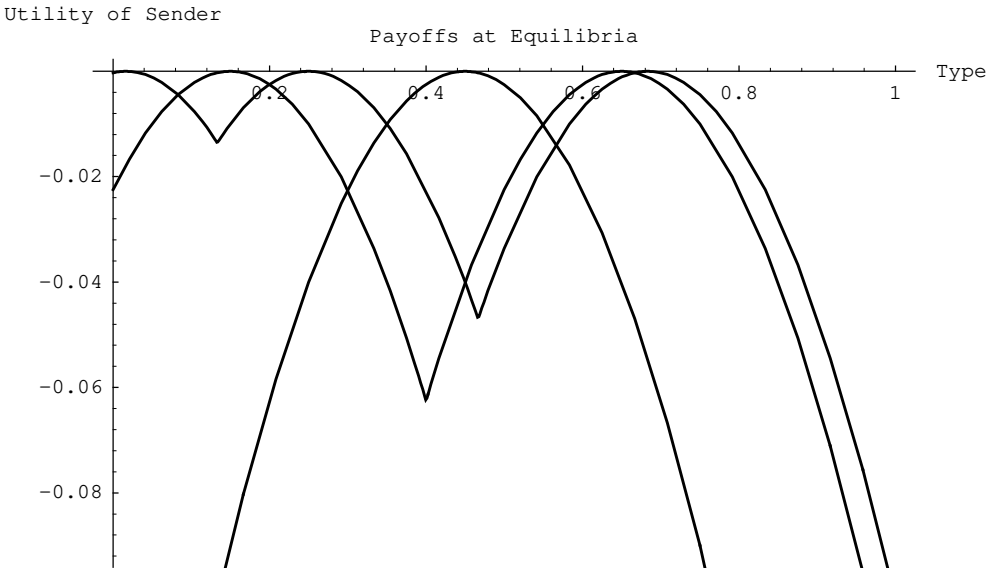
```
{-Graphics-, -Graphics-, -Graphics-}
```

```
parts
```

```
{{0., 1.}, {0., 0.4, 1.}, {0., 0.133333, 0.466667, 1.}}
```

In the above, **parts** represents the three different partitions of [0,1] that are equilibria. One of these is the uninformative partition {[0,1]}. The second has two elements, {[0,.0.4],[0.4,1]}. The third has three elements , {[0,.0.1333],[0.1333,.46667],[[0.46667,1]}.

```
Show[graphs,
AxesLabel->{"Type","Utility of Sender"},
PlotLabel->"Payoffs at Equilibria",
DisplayFunction->$DisplayFunction]
```

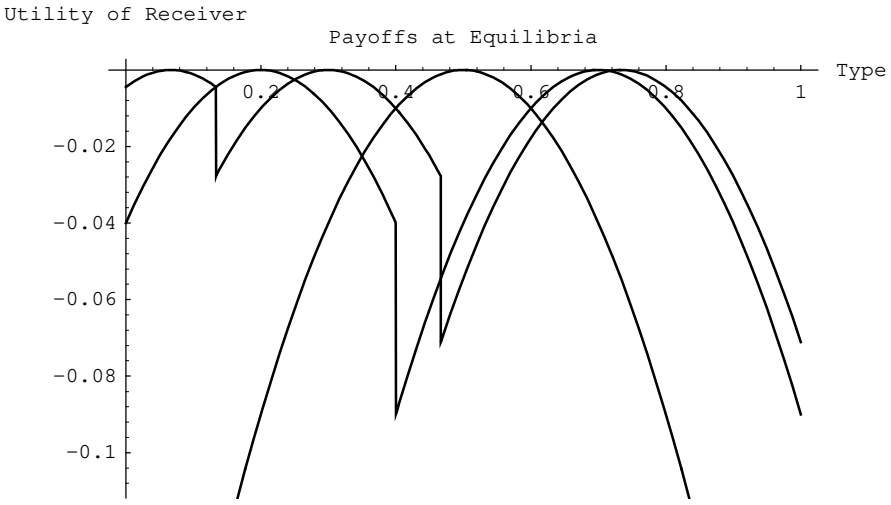



```
-Graphics-
```

This graph shows the utility of the Sender as a function of the state. The curve with one peak involves the no-communication partition {0,1}. The other two curves represent the two-element partition {0,.4,1} and three-element partition {0,.133,.4667,1} respectively.

Notice for the two and three element partitions the curves are continuous where they meet . This is due to the indifference condition that we used to solve this problem.

```
graphr=Table[Plot[Ur[y[parts[[n]],m],m],
   {m,0,1},PlotPoints->80,DisplayFunction->Identity],
   {n,1,3}];
Show[graphr,
AxesLabel->{"Type","Utility of Receiver"},
PlotLabel->"Payoffs at Equilibria",
DisplayFunction->$DisplayFunction]
```




```
-Graphics-
```

This graph shows the utility of the Receiver as a function of the state. Now notice the jump of the curve between elements. This represents a discontinuity that exists because the Receiver is not aware when the state is on the border of two elements, thus he need not be indifferent.

■ Example 4: If interests coincide at a point, does that imply full disclosure everywhere?

■ Preliminaries

In this second example we consider a variation on the standard Crawford-Sobel model in which the interests of the Sender and the Receiver do coincide, although at only one point. In the above examples, the interests of the two parties never coincided. A slight modification of the payoff functions ensures that at **m**=0 the two sides have identical interests. An interesting question to investigate is what happens as n, the number of partition elements, increases without bound.

■ Payoff functions

The utility functions are now

```
Us[y_,m_,b_]:=- (y-m -b m)^2;
Ur[y_,m_]:=-(y-m)^2;
{Us[y,0,b],Ur[y,0]}
```

```
{-y^2, -y^2}
```

Clearly, the players' preferences coincide at **m**=0.

The Sender's most preferred action and corresponding utility are

```
Sendersbest=Solve[D[Us[ym,m,b],ym]==0,ym]
Us[ym,0,b]/. Sendersbest
```

```
{{ym -> m + b m}}
```

```
{-(m + b m)^2}
```

```
Receiversbest=Solve[D[Ur[ym,m],ym]==0,ym]
Us[ym,0,b]/.Receiversbest
```

```
{{ym -> m}}
```

```
{-m^2}
```

■ Indifference Conditions

Now the conditions that generate a partition are for type a[i] to be indifferent between claiming to be in **[a[i-1], a[i]]** and claiming to be in **[a[i], a[i+1]]** .

```
-((a[i-1]+a[i])/2 -a[i] -b a[i])^2==
-((a[i]+a[i+1])/2 -a[i] -b a[i])^2
```

which can be simplified to

```
-(a[i-1]/2 -(2 b+1)a[i]/2)^2==
-(a[i]/2 -(2 b+1)a[i]/2)^2
```

Notice this is a quadratic and is of the form p^2 -q^2==0 which yields the root p+q=0 because p-q.

```
Simplify[(a[i-1]/2 -(2 b+1)a[i]/2)+
(a[i+1]/2 -(2 b+1)a[i]/2)]
```

$$\frac{a[-1+i]-2\,a[i]-4\,b\,a[i]+a[1+i]}{2}$$

■ Solving the indifference conditions

We solve the indifference conditions by using the function **values,** which then calls **PartitionEq.** We send **PartitionEq** the parameter **n** and the reduced indifference condition **(a[i+1]+a[i-1])/2 -a[i]-2 b a[i]==0**. We must also inform **PartitionEq** what variables are used in the indifference condition so we send it **{a,b,i}**. Since **PartitionEq** sometimes gives warning messages (see Package), we must prune it using **Select**.

```
values[n_]:=Select[PartitionEq[n,.03,,
(a[i+1]+a[i-1])/2 -a[i]-2 b a[i]==0,{a,b,i}],
Head[#1]===List &][[1]]

values[4]
```

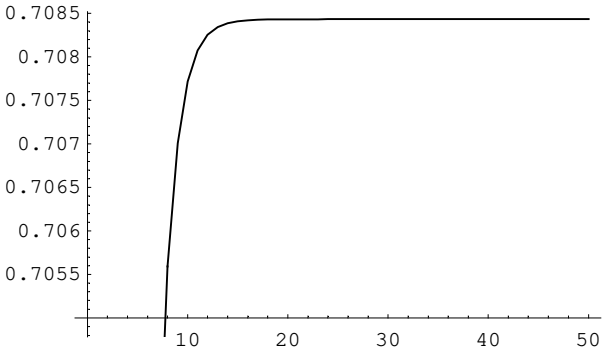
```
{0., 0.189103, 0.400898, 0.660801, 1.}
```

■ Asymptotic Results

At the beginning of this section, we mentioned that an unanswered question was what happened as N increases. Does full disclosure occur over the entire interval of types rather than in an neighborhood of zero? A sufficient condition for this not to occur is that the last partition has strictly positive length as N increases. We store the position of a[N-1] in **soltable**. As we try large values of N we see that a[N-1] converges.

```
soltable[i_]:=Block[{j},
     v=Table[values[j],{j,1,i}];
     Return[Table[v[[j]][[Length[v[[j]]]-1]],{j,1,i}]]];
soltable[50]
ListPlot[%,PlotJoined->True]
```

```
{0., 0.471698, 0.606685, 0.660801, 0.685307, 0.697013, 0.702747,
  0.70559, 0.707009, 0.707719, 0.708074, 0.708252, 0.708342,
  0.708387, 0.708409, 0.708421, 0.708426, 0.708429, 0.708431,
  0.708431, 0.708432, 0.708432, 0.708432, 0.708432, 0.708432,
  0.708432, 0.708432, 0.708432, 0.708432, 0.708432, 0.708432,
  0.708432, 0.708432, 0.708432, 0.708432, 0.708432, 0.708432,
  0.708432, 0.708432, 0.708432, 0.708432, 0.708432, 0.708432,
  0.708432, 0.708432, 0.708432, 0.708432, 0.708432, 0.708432,
  0.708432}
```




```
-Graphics-
```

How the Functions Work

Given a set of partitions that are equilibrium partitions, the function, **EqBeliefs,** can be used to calculate the conditional beliefs of the Receiver that exist in equilibrium for each message in each equilibrium set. Thus, if we had the following equilibrium partitions and probability function,

```
samplepartitions={{{1,2,3},{4}},{{1,2,3,4}}};
Us[y_,m_,b_]:=- (y-m-b)^2;
Ur[y_,m_]:=-(y-m)^2;Prob[x_]:=.25
```

iwe could calculate the beliefs regarding each of the underlying states when the different messages occur under the different partitions.

```
EqBeliefs[samplepartitions,Prob]
```

```
{{{0.333333, 0.333333, 0.333333, 0}, {0, 0, 0, 1.}},
  {{0.25, 0.25, 0.25, 0.25}}}
```

The function **EqBeliefs** is constructed from several straightforward calculations which depend on any prespecified probability function, **Prob.** The most primitive of these functions is **aggregator** which determines the probability of sets occurring. Thus, if one such set is **{1,2,3},** the probability of such a message is

```
aggregator[{1,2,3},Prob]
```

```
aggregator[{1, 2, 3}, Prob]
```

The function is defined in the following way

```
aggregator[x_,Prob_] := Apply[Plus, Prob /@ x]
```

With this function, it is possible to build conditional probabilities of states, given messages, as well as the collection of such conditional probabilties:

```
CondProb[state_, message_, Prob_] :=
  aggregator[Intersection[
  state, message],Prob]/aggregator[message,Prob]

 CondProbSet[S_,message_,Prob_] :=
    Table[CondProb[{s},message,Prob], {s,1,Length[S]}]
```

Thus the conditiional probability of state 1,**{1},** given the message of **{1,2,3},** the is

```
CondProb[{1},{1,2,3},Prob]
```

```
0.333333
```

To consider all states given **{1,2,3}** we would use

```
CondProbSet[{1,2,3,4},{1,2,3},Prob]
```

```
{0.333333, 0.333333, 0.333333, 0}
```

EqBeliefs performs this calculation for each member of the given partition.

```
EqBeliefs[k_,Prob_] :=
     Table[Table[CondProbSet[Flatten[k[[1]]],
     k[[j]][[i]],Prob],
    {i, 1, Length[k[[j]]]}], {j, 1, Length[k]}]
```

Thus, if only one partition is an equilibrium partition, say,**{{1},{1,2,3}}** , then the calculations are

```
EqBeliefs[{{{1},{2,3,4}}},Prob]
```

```
{{{1., 0, 0, 0}, {0, 0.333333, 0.333333, 0.333333}}}
```

We now turn to the task of computing what the equilbrium partitions are using **EqMessages.** The function **EqMessages** is composed of two basic operations: first, the function determines the candidates of sets of subsets of the message space that are potential equlibria. After selecting candidates, **EqMessages** uses the function **IsEquilibrium** to determine whether each candidate is an equilibrium.
The arguments of **EqMessages** include the number of states, the number of actions, the measure of the difference in preferences, the utility functions of the Receiver and Sender, and the underlying probability function defined on states. We will use four states, four actions, the utility functions used in our earlier discussion, and the constant probability function.

```
Us[y_,m_,b_]:=- (y-m-b)^2;
Ur[y_,m_]:=-(y-m)^2;Prob[x_]:=.25
```

These functions together with the number of states are used to create the basic representation of outcomes for both Receiver and Sender.

```
RUtils[A_, S_, Ur_] :=
  Table[Ur[a, s], {a, 1, Length[A]}, {s, 1, Length[S]}]


SUtils[A_, s_, b_,Us_] := Table[Us[a, s, b],
{a, 1, Length[A]}]
```

For the Receiver, we would have

```
MatrixForm[RUtils[Range[4],Range[4],Ur]]
```

```
0    -1   -4   -9
-1   0    -1   -4
-4   -1   0    -1
-9   -4   -1   0
```

Because the Sender knows what the state is when choosing a message, the basic representation of the Sender's preferences is state-dependent.

```
SUtils[Range[4],1,2.0,Us]
```

```
{-4., -1., 0., -1.}
```

The table of overall Sender's values would be

```
MatrixForm[Table[SUtils[Range[4],s,2.0,Us],{s,1,4}]]
```

```
-4.    -1.    0.     -1.
-9.    -4.    -1.    0.
-16.   -9.    -4.    -1.
-25.   -16.   -9.    -4.
```

Thus we are working with the setting in which preferences are not aligned.

We first discuss the function **IsEquilibrium**. It will help to begin with a candidate for an equilbrium message set , such as **{{1},{1,2,3}}**. With each message in the set, condidtional probabilities of each of the states can be formed in the manner described earlier, and from this calculation we can determine the expected utility of each action for a particular message.

```
ExpUtilR[A_, S_, message_,Prob_,Ur_] :=
  Transpose[CondProbSet[S,message,Prob]] .
  RUtils[A, S,Ur]
```

Thus, for the second message, **{2,3,4}**, we have

```
ExpUtilR[Range[4],Range[4],{2,3,4},Prob,Ur]
```

```
{-4.66667, -1.66667, -0.666667, -1.66667}
```

Thus the best response for Receiver given the message **{1,2,3}**, would be action 2, since it yields the highest utility. This is directly established by

```
BestResponse[A_, S_, message_,Prob_,Ur_]:=
  Flatten[
    Position[
        ExpUtilR[A, S, message,Prob,Ur],
        Max[ExpUtilR[A, S, message,Prob,Ur]]]
]
```

The function locates the positions where a maximum occurs and then returns the appropriate action:

```
BestResponseSet[A_, S_, PART_,Prob_,Ur_] :=
  Table[BestResponse[A, S, PART[[i]],Prob,Ur],
    {i, 1, Length[PART]}]

BestResponseSet[Range[4],Range[4],{{1},{2,3,4}},Prob,Ur]
```

```
{{1}, {3}}
```

Thus we have the collection of best responses for the Receiver in this setting, given the underlying set of possible messages. However, we do not know that the Sender will, in fact, send the appropriate message, when the Sender knows how the

Receiver will react. To have an equilibrium, it must be the case that the actions the Sender attempts to induce are in fact the actions the Receiver will take. Thus, given a Sender knows a particular state, we need to examine what action, from those available in the best response set of the Receiver, that the Sender will attempt to induce. This is accomplished by the function

```
SendersBest[A_, S_, s_, PART_, b_,Ur_,Us_,Prob_] :=
  coord[Flatten[BestResponseSet[A, S, PART,Prob,Ur]]] /@
   Position[
       SUtils[Flatten[BestResponseSet[A, S,PART,Prob,Ur]],
    s, b,Us],
       SMax[BestResponseSet[A, S, PART,Prob,Ur], s, b,Us]
       ]
```

The function first determines the **`BestResponseSet`** of the Receiver,then determines the utility for the Sender for each of these responses, then determines which of these utilities is the maximum in the set of utilities, and finally picks out those Receiver's actions that yield this utility. Consider how this function builds up. First consider

```
BestResponseSet[Range[4],Range[4],{{1},{2,3,4}},Prob,Ur]
```

```
{{1}, {3}}
```

Then for a particular state, say 3, the Sender's possible utilties are

```
SUtils[Flatten[BestResponseSet[Range[4],Range[4],{{1},
{2,3,4}},Prob,Ur]],1,2.0,Us]
```

```
{-4., -1.}
```

```
SMax[A_, s_, b_,Us_] := Max[SUtils[Flatten[A], s, b,Us]]
```

```
SMax[BestResponseSet[Range[4],Range[4], {{1},
{2,3,4}},Prob,Ur], 1, 2.0,Us]
```

```
-1.
```

```
coord[set_] := Function[x, set[[x]]]
```

Thus, the Sender would like to see action 3 taken, i.e.,

```
SendersBest[Range[4],Range[4],1,{{1},{2,3,4}},2.0,Ur,
Us,Prob]
```

```
{{3}}
```

Thus the Sender's desired action and Receiver's desired action do not match, given the partition. We check this out with the function Match

```
Match[A_, S_, s_, PART_,b_,Ur_,Us_,Prob_] :=
  Intersection[
    SendersBest[A, S, s, PART, b,Ur,Us,Prob][[1]],
   {BestResponseSet[A, S, PART,Prob,Ur][[s]]}[[1]]]
```

```
Match[Range[4],Range[4],1,{{1},{2,3,4}},2.0,Ur,Us,Prob]
```

```
{}
```

When any particular state leads to **Match** being the null set, there cannot be an equilibrium, which we find out by using **IsEquilbrium**:

```
IsEquilibrium[A_, S_, PART_, b_,Ur_,Us_,Prob_] :=
  Module[{a1, b1, c1, d1, e1,f1,g1,h1},
    a1 = 1; b1 = b; c1 = A; d1 = S; e1 = PART;
    f1=Ur;g1=Us;h1=Prob;
        While[a1 <= Length[S],
        If[Match[c1, d1, a1, e1, b1,f1,g1,h1] == {},
        Return[{}]]; a1++];
   Return[e1]]
```

that is

```
IsEquilibrium[Range[4],Range[4],{{1},{2,3,4}},2.0,Ur,
Us,Prob]
```

```
{}
```

The final element of the calculations is to determine the set of candidates for equilibrium sets of messages. The fundamental part of the calculation is a function, **partition**, which for a given set and a number, n, will determine all partitions which contain n subsets of the set. For example

```
partition[{1,2,3,4}, 3]
```

```
partition[{1, 2, 3, 4}, 3]
```

The function is defined recursively with

```
partition[set_, 1] := {set}

partition[set_, 2] := Table[part[i][set],
{i, 1, Length[set] - 1}]

part[i_] := Function[x, {Take[x, i],
Take[x, -(Length[x] - i)]}]
```

When the partition is of size one the function is obvious. For partitions of size 2 the determination depends on a pure function **part[i_]** which will divide a set into its first i and remaining elements. For example,

```
part[2][{1,2,3,4}]
```

```
{{1, 2}, {3, 4}}
```

The collection over i will be the partitions of size 2.

```
Table[part[i][{1,2,3,4}], {i, 1, Length[{1,2,3,4}] - 1}]
```

```
{{{1}, {2, 3, 4}}, {{1, 2}, {3, 4}}, {{1, 2, 3}, {4}}}
```

We now consider the recursive formula which is

```
partition[set_, n_] :=
  partition[set, n] =
   Flatten[
    Table[
        Table[Insert[
                 partition[Take[set, -(Length[set] - i)],
                        n - 1][[j]],
              Take[set, i], 1],
         {j, 1, Length[partition[Take[
                   set, -(Length[set] - i)], n - 1]]}],
     {i, 1, Length[set] - (n - 1)}],
    1]
```

The recursion works on the principle that if we take increasingly large subsets of the set to be partitioned and append all partitions of size n-1, then we ultimately have all the partitions of size n. Thus, suppose we start with the set, **{1,2,3,4},** and wish to find the partitions of size 3. We work succesively with the sets, **{1}** and **{1,2},** and append to them subsets of the remaining numbers of size 2. Thus, if we append **{1}** to

```
partition[{2,3,4},2]
```

```
{{{2}, {3, 4}}, {{2, 3}, {4}}}
```

we will have the desired partitions with {1} as the first subset of the partition. When we look at both sets {1} and {1,2}, we have

```
partition[{1,2,3,4},3]
```

```
{{{1}, {2}, {3, 4}}, {{1}, {2, 3}, {4}}, {{1, 2}, {3}, {4}}}
```

Thus, we have a method of determining the set of all **connected** possible partitions of any set.

We will now enumerate some partitions of the set of types, and check whether sending a particular subset from those partitions is an equilibrium. In doing so, we will not enumerate all possible subsets of the set of types. We will restrict our search to "connected" partitititions of the set of types.

Formally, a partition P of the set of types is said to be connected if, for every partition element Pi , if two non-adjacent types tj and tk are in Pi, then all intermediate types are also in Pi. For instance if the set of types is {1,2,3,4} we will ignore partitions of the form {{1,3},{2},{4}} because they are disconnected. This is because {2} is between {1} and {3} so the partition {1,3} has a "hole" at {2}. Partitions such as {{1,2,3},{4}} or {{1,2,3},{3,4}}are, by contrast, connected. Connected partitions such as {{1,2,3},{4}} do not overlap; others such as {{1,2,3},{3,4}} do overlap in the sense that adjacent elements in the partition share one or more states.

If preferences satisfy the "single-crossing" property then disconnected partitions cannot be equilibria. Crawford and Sobel assume the "single-crossing property", and this is satisfied by the utility functions we have specified. The single-crossing

property is used in the economics literature to study optimal taxation, signalling and incentive contracts. Intuitively, the property implies that "higher" types care more about the action than "lower" types: in our problem, the marginal utility of the action increases with the type.

The set of all partitions we have derived is not the set of all combinations of message sets that are candidates for equilbria. It may be that **overlapping** connected partitions constitute possible equilibrium message sets. For example, we may have, in addition to the partition{{1},{2},{3,4}}, the sets, {{1,2},{2,3},{3,4}} as possible message sets in equilibrium. Not only is the above set a viable message set but also combinations of this set and the original partition are. So, for example, the set {{1},{2},{2,3,4}} is a viable message set. We derive the set of all possibilities in two steps. First, to find the message set in which the last member of a subset of the initial partition is added to the next element of the partition, we employ two functions, overlapping and augment.

```
overlapping[par_] := Fold[augment, {par[[1]]}, Rest[par]]

augment = Join[#1, {Join[{#1[[-1]][[-1]]}, #2]}] &
```

```
Join[#1, {Join[{#1[[-1]][[-1]]}, #2]}] &
```

If we apply **augment** to the two sets {{1}} and {2} we get

```
augment[{{1}},{2}]
```

```
{{1}, {1, 2}}
```

If we combine the result with the remaining set, {3,4}, we get

```
augment[augment[{{1}},{2}],{3,4}]
```

```
{{1}, {1, 2}, {2, 3, 4}}
```

The repeated use of augment to any partition can be accomplished through the function **overlapping**.

```
overlapping[{{1},{2},{3,4}}]
```

```
{{1}, {1, 2}, {2, 3, 4}}
```

To find the set of all combinations of these sets, we use the functions

```
completegroup[set_] :=
  create[Transpose[{set, overlapping[set]}]]

create[setoverlap_] :=
  Union[Fold[add, List /@ setoverlap[[-1]],
  Reverse[Drop[setoverlap, -1]]]]

add = Flatten[Table[Table[Insert[#1[[i]], #2[[j]], 1],
      {i, 1, Length[#1]}], {j, 1, 2}], 1] &
```

```
Flatten[Table[Table[Insert[#1[[i]], #2[[j]], 1], {i, 1, Length[#1]}],
    {j, 1, 2}], 1] &
```

```
add[{{a1},{b1}},{c1,d1}]
```

```
{{c1, a1}, {c1, b1}, {d1, a1}, {d1, b1}}
```

```
add[%,{e1,f1}]
```

```
{{e1, c1, a1}, {e1, c1, b1}, {e1, d1, a1}, {e1, d1, b1}, {f1, c1, a1},
  {f1, c1, b1}, {f1, d1, a1}, {f1, d1, b1}}
```

The function **add**, then is a method of augmenting pairs to create triples with each element of the pair positioned in front of the previous pair.

```
create[{{a1,b1},{c1,d1},{e1,f1}}]
```

```
{{a1, c1, e1}, {a1, c1, f1}, {a1, d1, e1}, {a1, d1, f1}, {b1, c1, e1},
  {b1, c1, f1}, {b1, d1, e1}, {b1, d1, f1}}
```

The function **add** is applied, the same sets are created, but the ordering is reversed for each of the sets. Note that the set **{{a1,b1},{c1,d1},{e1,f1}}** is the transpose of **{{a1,c1,e1},{b1,d1,f1}}.** Using this observation we apply **create** to the original partition and its overlap,

```
completegroup[{{1},{2},{3,4}}]
```

```
{{{1}, {2}, {3, 4}}, {{1}, {2}, {2, 3, 4}}, {{1}, {1, 2}, {3, 4}},
  {{1}, {1, 2}, {2, 3, 4}}}
```

The function **EqMessages** has been explained in terms of its two major components. The function, **completegroup**, is used in determining the set of all possible candidates for equilibrium messages, and the function **IsEquilibrium** is used to evaluate each of these candidates.

The package, **strategic**, provides a useful vehicle for examining sender-receiver games, and thus provides a first applications to games of incomplete information, i.e. games with information asymmetry. We appreciate Jack Stecher's comments on this manuscript.

The Package

```
BeginPackage["cspackage`"]

PartitionEq::usage="PartitionEq[n,b] calculates
the n partitions for a continuum of types on [0,1] given a preference parameter b.
PartitionEq[n,b,us] calculates the n partitions for a continuum of types on [0,1] given a
preference parameter b and a utility function us for  the  Sender. PartitionEq[n,b,,eqopt]
calculates the n partitions for a continuum of types on [0,1] given a preference parameter b
uses a reduced difference equation eqopt to simplify the calculation.  Defaults are uniform
prior distribution on [0,1] and quadratic preferences if no other utility function is given."

EqMessages::usage="EqMessages[states,actions,b,Ur,Us,Prob]
calculates the sets of equilibrium messages for the
case of finitely many states, drawn from the set
{1,states}, on the set {1,actions}, where actions and states are positive integers.  Ur and Us
are the utility functions of the Receiver and the Sender, with b a non-zero measure of the
divergence in their preferences. Prob is the prior probability distribution on the
set{1,states}. "

EqBeliefs::usage="EqBeliefs[k,Prob] calculates the equilibrium beliefs, for the case of
discretely many states, for sets of equilibrium messages found by the function EqMessages, and
a prior probability distribution Prob on the set {1,states} where states is a positive integer."

ReActions::usage="ReActions[A, S, PART,Ur,Prob] :=calculates  the sets of optimal actions from
the set A by the Receiver for the case of discretely many states (drawn from the set S), given
a partition (PART) of S and a utility function Ur for the Receiver when Prob is the prior
probability  distribution on S. "

Begin["`Private`"]

EqBeliefs[k_,Prob_] :=
     Table[Table[CondProbSet[Flatten[k[[1]]],
     k[[j]][[i]],Prob],
    {i, 1, Length[k[[j]]]}], {j, 1, Length[k]}]

CondProbSet[S_,message_,Prob_] :=
    Table[CondProb[{s},message,Prob], {s,1,Length[S]}]

CondProb[state_, message_, Prob_] :=
  aggregator[Intersection[
  state, message],Prob]/aggregator[message,Prob]

aggregator[x_,Prob_] := Apply[Plus, Prob /@ x]

EqMessages[actions_, states_,b_,Ur_,Us_,Prob_] :=
  Join[{{Range[states]}}, Complement[Flatten[
    Table[
         Table[IsEquilibrium[Range[actions],                        Range[states],Flatten[
             completegroup /@
             partition[Range[states],i], 1][[j]],                    b,Ur,Us,Prob],
          {j, 1, Length[Flatten[completegroup /@
partition[Range[states], i], 1]]}],
      {i, 2, states}], 1],
   {{}}]]
IsEquilibrium[A_, S_, PART_, b_,Ur_,Us_,Prob_] :=
  Module[{a1, b1, c1, d1, e1,f1,g1,h1},
    a1 = 1; b1 = b; c1 = A; d1 = S; e1 = PART;
    f1=Ur;g1=Us;h1=Prob;
        While[a1 <= Length[S],
        If[Match[c1, d1, a1, e1, b1,f1,g1,h1] == {},
        Return[{}]]; a1++];
   Return[e1]]
Match[A_, S_, s_, PART_,b_,Ur_,Us_,Prob_] :=
```

```
  Intersection[
    SendersBest[A, S, s, PART, b,Ur,Us,Prob][[1]],
   ResponseState[A, S, PART,Ur,Prob][[s]]]

SendersBest[A_, S_, s_, PART_, b_,Ur_,Us_,Prob_] :=
  coord[Flatten[BestResponseSet[A, S, PART,Prob,Ur]]] /@
   Position[
       SUtils[Flatten[BestResponseSet[A, S,PART,Prob,Ur]],                                    s, b,Us],
       SMax[BestResponseSet[A, S, PART,Prob,Ur], s, b,Us]
       ]

coord[set_] := Function[x, set[[x]]]
BestResponseSet[A_, S_, PART_,Prob_,Ur_] :=
  Table[BestResponse[A, S, PART[[i]],Prob,Ur],
    {i, 1, Length[PART]}]

BestResponse[A_, S_, message_,Prob_,Ur_]:=
  Flatten[
    Position[
        ExpUtilR[A, S, message,Prob,Ur],
        Max[ExpUtilR[A, S, message,Prob,Ur]]]
]

ExpUtilR[A_, S_, message_,Prob_,Ur_] :=
  Transpose[CondProbSet[S,message,Prob]] .
  RUtils[A, S,Ur]

RUtils[A_, S_, Ur_] :=
  Table[Ur[a, s], {a, 1, Length[A]}, {s, 1, Length[S]}]

SUtils[A_, s_, b_,Us_] := Table[Us[a, s, b],
{a, 1, Length[A]}]

SMax[A_, s_, b_,Us_] := Max[SUtils[Flatten[A], s, b,Us]]

RBestActions[A_, S_, PART_,Prob_,Ur_] :=
  Flatten[Table[Table[BestResponse[A, S,
  PART[[i]],Prob,Ur],
     {j, 1, Length[PART[[i]]]}], {i, 1, Length[PART]}], 1]

ResponseState[A_,S_,PART_,Ur,Prob_]:=
    Flatten[Table[Table[BestResponse[A,S,PART[[i]],
    Prob,Ur],{j,1,Length[PART[[i]]]}],{i,1,Length[
    PART]}],1]

completegroup[initpart_] :=
 create[Transpose[{initpart, overlapping[initpart]}]]

create[setoverlap_] :=
  Union[Fold[add, List /@ setoverlap[[-1]],
  Reverse[Drop[setoverlap, -1]]]]

add = Flatten[Table[Table[Insert[#1[[i]], #2[[j]], 1],
      {i, 1, Length[#1]}], {j, 1, 2}], 1] &

overlapping[par_] := Fold[augment, {par[[1]]}, Rest[par]]

augment = Join[#1, {Join[{#1[[-1]][[-1]]}, #2]}] &

partition[set_, 1] := {set}

partition[set_, 2] := Table[part[i][set],
    {i, 1, Length[set] - 1}]

partition[set_, n_] :=
  partition[set, n] =
   Flatten[
    Table[
```

```
        Table[Insert[
                 partition[Take[set, -(Length[set] - i)],
                           n - 1][[j]],
               Take[set, i], 1],
         {j, 1, Length[partition[Take[
                   set, -(Length[set] - i)], n - 1]]}],
     {i, 1, Length[set] - (n - 1)}],
    1]
part[i_] := Function[x, {Take[x, i], Take
    [x, -(Length[x] - i)]}]
ReActions[A_, S_, PART_,Ur_,Prob_] :=
  Table[BestResponse[A, S, PART[[i]],Prob,Ur],
    {i, 1, Length[PART]}]
boundcond[us_]:=Block[{y,m,a},
  ExpUtil=Integrate[us[y,m,0],{m,a[i],a[i+1]}];
  foc=D[ExpUtil,y]==0;
  y[i_]:=Evaluate[y/.Flatten[Simplify[Solve[foc,y]]]];
 Return[ us[y[i],a[i],b]==us[y[i-1],a[i],b]]
  ];
PartitionEq[n_,bb_,us_:1,eqopt_:1]:=Block[
{ans1,ans2,ans,eq,f,eqns,i,j,g},
If[n==1,Return[{0.,1.}] ];
If[eqopt===1,
  If[us===1,
    eq[i_]:=a[i+1]==2 a[i]-a[i-1]+4 bb,
    eq[i_]:=Evaluate[boundcond[us]/.b->bb];
  ];,
eq[i_]:=Evaluate[eqopt/.b->bb] ];
eqns=Join[Table[eq[i],{i,1,n-1}],{a[0]==0,a[n]==1}];
g=NSolve[eqns,Table[a[i],{i,0,n}]];
ans1=Table[Table[a[i],{i,0,n}]/.g[[j]],
    {j,1,Length[g]}];
f[a_]:=Apply[And,Table[a[[i]]>a[[i-1]],{i,2,Length[a]}]];
ans=Select[ans1,f];
ans2=Select[ans1,Not[Apply[And,Map[NumberQ,#1]]] &];
If[Length[ans2]>0,ans=Join[ans,
{"Warning! Some partitions might not be increasing."},
ans2] ];
Return[ans];
];
 End[]
 EndPackage[]
```

■ References

Crawford V. and Sobel J., Strategic Information Transmission. *Econometrica* 50: 1982

Dickhaut J. and Kaplan T. A Program for Finding Nash Equilibria. The Mathematica Journal 1(4) 1991

Myerson R., Game Theory. Cambridge, Mass: Harvard University Press 1991

Harsanyi J., Games of Incomplete Information Played by 'Bayesian' Players. Management Science 14: 1967

Tirole J., The Theory of Industrial Organization. Boston, Mass: M.I.T. Press 1989