

Topics in Social Software
Information in Strategic Situations

Eric Pacuit

April 15, 2004

Department of Computer Science
City University of New York Graduate Center

www.cs.gc.cuny.edu/~epacuit

epacuit@cs.gc.cuny.edu

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations
- A Theory of Correctness of Social Procedures

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- Mathematical Models of Social Situations
- A Theory of Correctness of Social Procedures
- Designing Social Procedures

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Research in Social Software can be divided into three different but related categories:

- **Mathematical Models of Social Situations**
- A Theory of Correctness of Social Procedures
- Designing Social Procedures

Introduction: What is social software

Social software is an **interdisciplinary research program** that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Introduction: What is social software

Social software is an **interdisciplinary research program** that combines mathematical tools and techniques from game theory and computer science in order to analyze and design social procedures.

Related Fields:

- Computer Science (algorithms, logic of programs, ...),
- Philosophical Logic (epistemic logic, deontic logic, ...),
- Game Theory (mechanism design, ...),
- Artificial Intelligence (multi-agent systems, ...)

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and **computer science** in order to analyze and design social procedures.

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and **computer science** in order to analyze and design social procedures.

Just as computer programs have logical properties which are analyzed by means of appropriate logics of programs, social procedures also have logical properties which can be analyzed with the appropriate logical tools.

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and **computer science** in order to analyze and design social procedures.

Just as computer programs have logical properties which are analyzed by means of appropriate logics of programs, social procedures also have logical properties which can be analyzed with the appropriate logical tools.

- Game Logic (Parikh, Paily)
- Coalitional Logic (Paily, Goranko)
- Game Algebra (van Benthem, Goranko, Venema, Paily)

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design **social** procedures.

Introduction: What is social software

Social software is an interdisciplinary research program that combines mathematical tools and techniques from game theory and computer science in order to analyze and design **social** procedures.

Social procedures occur at two levels

1. Individual algorithms set up by a **group** of agents
(taking a train, dropping a class, ...)
2. Algorithms that require a **group** even in the execution
(voting procedures, fair division algorithms, piano duets, ...)

Introduction: Example

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Suppose Ann and Bob are dividing three goods: A , B , and C .

Step 1. Both Ann and Bob divide 100 points the three goods.

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Suppose Ann and Bob are dividing three goods: A , B , and C .

Step 1. Both Ann and Bob divide 100 points the three goods.

| Item | Ann | Bob |
|--------------|-----|-----|
| A | 5 | 4 |
| B | 65 | 46 |
| C | 30 | 50 |
| Total | 100 | 100 |

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 2. The agent who assigns the most points receives the item.

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 2. The agent who assigns the most points receives the item.

| Item | Ann | Bob |
|--------------|-----|-----|
| <i>A</i> | 5 | 4 |
| <i>B</i> | 65 | 46 |
| <i>C</i> | 30 | 50 |
| Total | 100 | 100 |

Introduction: Example

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 2. The agent who assigns the most points receives the item.

| Item | Ann | Bob |
|--------------|-----------|-----------|
| <i>A</i> | 5 | 0 |
| <i>B</i> | 65 | 0 |
| <i>C</i> | 0 | 50 |
| Total | 70 | 50 |

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|-----------|-----------|
| <i>A</i> | 5 | 0 |
| <i>B</i> | 65 | 0 |
| <i>C</i> | 0 | 50 |
| Total | 70 | 50 |

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|-----------|-----------|
| <i>A</i> | 0 | 4 |
| <i>B</i> | 65 | 0 |
| <i>C</i> | 0 | 50 |
| Total | 65 | 54 |

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|-----------|-----------|
| <i>A</i> | 0 | 4 |
| <i>B</i> | 65 | 0 |
| <i>C</i> | 0 | 50 |
| Total | 65 | 54 |

Let p be the proportion of item B that Ann will keep. Then

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|-----------|-----------|
| A | 0 | 4 |
| B | 65 | 0 |
| C | 0 | 50 |
| Total | 65 | 54 |

Let p be the proportion of item B that Ann will keep. Then

$$65p = 100 - 46p$$

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|-----------|-----------|
| <i>A</i> | 0 | 4 |
| <i>B</i> | 65 | 0 |
| <i>C</i> | 0 | 50 |
| Total | 65 | 54 |

yielding $p = 100/111 = 0.9009$

Introduction: Adjusted Winner

Adjusted winner is an algorithm to fairly distribute divisible goods among two people. [BT]

Step 3. Equitability adjustment:

| Item | Ann | Bob |
|--------------|---------------|---------------|
| <i>A</i> | 0 | 4 |
| <i>B</i> | 58.559 | 4.559 |
| <i>C</i> | 0 | 50 |
| Total | 58.559 | 58.559 |

yielding $p = 100/111 = 0.9009$

Introduction: Example

- Adjusted Winner is *envy-free*, *equitable* and *efficient*. [BF]
- Can the agents improve their allocation by misrepresenting their preferences?

Introduction: Example

- Adjusted Winner is *envy-free, equitable and efficient*. [BF]
- Can the agents improve their allocation by misrepresenting their preferences?

Yes. However, *Unless an agent is certain that its preference is private and that its information about the other agent's preference is correct, then that agent is better off reporting its true preference.*

Introduction: Example

- Adjusted Winner is *envy-free, equitable and efficient*. [BF]
- Can the agents improve their allocation by misrepresenting their preferences?

Yes. However, *Unless an agent is certain that its preference is private and that its information about the other agent's preference is correct, then that agent is better off reporting its true preference.*



Introduction: Objective

Introduction: Objective

One of the central issues in social software and the focus of this thesis is the development of appropriate models of multi-agent interactive situations.

Introduction: Objective

One of the central issues in social software and the focus of this thesis is the development of appropriate models of multi-agent interactive situations.

The objective of this thesis is to develop logical systems that can be used to reason about multi-agent interactive situations relevant for the analysis of social procedures.

Introduction: Objective

One of the central issues in social software and the focus of this thesis is the development of appropriate models of multi-agent interactive situations.

The objective of this thesis is to develop logical systems that can be used to reason about multi-agent interactive situations relevant for the analysis of social procedures.

The role of information and its dynamics in strategic multi-agent situations is a central issue.

Introduction: Two Issues

Introduction: Two Issues

1. Knowledge, Actions and Obligations

Introduction: Two Issues

1. Knowledge, Actions and Obligations

An Agents choice of action depends on its current state of information. Certainly an agent cannot be faulted for not performing an action whose need it did not know about.

Introduction: Two Issues

1. Knowledge, Actions and Obligations

An Agents choice of action depends on its current state of information. Certainly an agent cannot be faulted for not performing an action whose need it did not know about.

2. Updating Information

Introduction: Two Issues

1. Knowledge, Actions and Obligations

An Agents choice of action depends on its current state of information. Certainly an agent cannot be faulted for not performing an action whose need it did not know about.

2. Updating Information

Just as registers in a computer are continually updated as a program is executed, the information of each agent changes as a social procedure is executed.

Introduction: ...And a Third (time permitting)

Introduction: ...And a Third (time permitting)

3. Strategic Voting

Introduction: ...And a Third (time permitting)

3. Strategic Voting

Game theory can be used to ensure that agents will perform the requisite task. But how do the agents *know* that they should follow the recommendations game theory offers?

Overview

- **Introduction**
- **Background: History Based Models**
- **First Issue: Knowledge Based Obligation ([PPC])**
- **Second Issue: The Logic of Communication Graphs ([PP])**
- **Third Issue: Strategic Voting ([CPP])**
- **Conclusions and Further Work**

Overview

- Introduction
- **Background: History Based Models**
- First Issue: Knowledge Based Obligation ([PPC])
- Second Issue: The Logic of Communication Graphs ([PP])
- Third Issue: Strategic Voting ([CPP])
- Conclusions and Further Work

Overview

- Introduction
- Background: History Based Models
- **First Issue: Knowledge Based Obligation ([PPC])**
- Second Issue: The Logic of Communication Graphs ([PP])
- Third Issue: Strategic Voting ([CPP])
- Conclusions and Further Work

Overview

- Introduction
- Background: History Based Models
- First Issue: Knowledge Based Obligation ([PPC])
- **Second Issue: The Logic of Communication Graphs ([PP])**
- Third Issue: Strategic Voting ([CPP])
- Conclusions and Further Work

Overview

- Introduction
- Background: History Based Models
- First Issue: Knowledge Based Obligation ([PPC])
- Second Issue: The Logic of Communication Graphs ([PP])
- **Third Issue: Strategic Voting ([CPP])**
- Conclusions and Further Work

Overview

- Introduction
- Background: History Based Models
- First Issue: Knowledge Based Obligation ([PPC])
- Second Issue: The Logic of Communication Graphs ([PP])
- Third Issue: Strategic Voting ([CPP])
- **Conclusions and Further Work**

Background: History Based Structures I

The semantics is based on the history based semantics of [PR] (1985, 2003).

- Let E be a fixed set of **events**.
- Elements of $E^* \cup E^\omega$ will be called **histories**. Three types: 1. local histories, 2. finite global histories and 3. infinite global histories.
- Given two histories H' and H , write $H \preceq H'$ to mean H is a **finite prefix** of H' .
- Define $\text{FinPre}(\mathcal{H}) = \{H \mid H \in E^*, H \preceq H', \text{ and } H' \in \mathcal{H}\}$.
- Given a history H , H_t is the finite prefix of H of length t .

Background: History Based Structures II

- A set $\mathcal{H} \subseteq E^* \cup E^\omega$ is called a **protocol** if \mathcal{H} is closed under FinPre.
- A **history based structure** based on a set of events E is a tuple $\langle \mathcal{H}, E_1, \dots, E_n \rangle$ where \mathcal{H} is a protocol and $E_i \subseteq E$ for each $i \in \mathcal{A}$.
- If $e \in E_i$ then it is possible for i to be aware of event e .

Background: History Based Structures II

- A set $\mathcal{H} \subseteq E^* \cup E^\omega$ is called a **protocol** if \mathcal{H} is closed under FinPre.
- A **history based structure** based on a set of events E is a tuple $\langle \mathcal{H}, E_1, \dots, E_n \rangle$ where \mathcal{H} is a protocol and $E_i \subseteq E$ for each $i \in \mathcal{A}$.
- If $e \in E_i$ then it is possible for i to be aware of event e .

Background: History Based Frames

- For each $i \in A$ define $\lambda_i : \text{FinPre}(\mathcal{H}) \rightarrow E_i^*$ to be the **local view function** of agent i .

Background: History Based Frames

- For each $i \in \mathcal{A}$ define $\lambda_i : \text{FinPre}(\mathcal{H}) \rightarrow E_i^*$ to be the **local view function** of agent i .

We can define for each **finite** $H \in \mathcal{H}$,

$$H \sim_i H' \text{ iff } \lambda_i(H) = \lambda_i(H')$$

- A **history based frame** is a tuple $\langle \mathcal{H}, E_1, \dots, E_n, \lambda_1, \dots, \lambda_n \rangle$.

Background: History Based Frames

- For each $i \in \mathcal{A}$ define $\lambda_i : \text{FinPre}(\mathcal{H}) \rightarrow E_i^*$ to be the **local view function** of agent i .

We can define for each **finite** $H \in \mathcal{H}$,

$$H \sim_i H' \text{ iff } \lambda_i(H) = \lambda_i(H')$$

- A **history based frame** is a tuple $\langle \mathcal{H}, E_1, \dots, E_n, \lambda_1, \dots, \lambda_n \rangle$.
- Assumption for this talk: $\lambda_i(H)$ is obtained by mapping each event "seen" by i into itself and each event not seen by i to the empty string (**perfect recall**)

Background: Temporal Epistemic Logic

- A formula $\phi \in \mathcal{L}_n^{KT}$ can have the following form

$$\phi := p \mid \neg\phi \mid \phi \wedge \psi \mid K_i\phi \mid \bigcirc\phi \mid \phi U\psi$$

- A **valuation function** is a function $V : \text{FinPre}(\mathcal{H}) \rightarrow 2^{\text{At}}$
- If $H \in \mathcal{H}$ is a **infinite** global history, then $H, t \models \phi$ is intended to mean ϕ is true at time t in history H :

1. $H, t \models \phi U\psi$ iff there exists $m \geq t$ such that $H, m \models \psi$ and for all l such that $t \leq l < m$, $H, l \models \phi$
2. $H, t \models K_i\phi$ iff for all $H' \in \mathcal{H}$ such that $H_t \sim_i H'_t$, $H', t \models \phi$.
- 2'. $H, t \models K_i\phi$ iff for all $m \geq 0$, for all $H' \in \mathcal{H}$ such that $H_t \sim_i H'_m$, $H', m \models \phi$

History Based Models: Truth

Given $\mathcal{M}_H = \langle \mathcal{H}, \lambda_1, \dots, \lambda_n, V \rangle$:

1. $H, k \models F\phi$ iff there exists $m > k$ such that $H, m \models \phi$
 2. $H, k \models K_i\phi$ iff for all $H' \in \mathcal{H}$ such that $H_k \sim_i H'_k$, $H', k \models \phi$.
- 2'. $H, k \models K_i\phi$ iff for all $m \geq 0$, for all $H' \in \mathcal{H}$ such that $H_k \sim_i H'_m$, $H'_m, m \models \phi$
- K_i is an **S5** operator for each i .
 - A sound and complete axiomatization for knowledge and time can be found in [HvdMV, 2003] (linear time), [vdMW, 2003] (branching time) using a slightly different framework.

History Based Models: Truth

Given $\mathcal{M}_H = \langle \mathcal{H}, \lambda_1, \dots, \lambda_n, V \rangle$:

1. $H, k \models F\phi$ iff there exists $m > k$ such that $H, m \models \phi$
 2. $H, k \models K_i\phi$ iff for all $H' \in \mathcal{H}$ such that $H_k \sim_i H'_k$, $H', k \models \phi$.
- 2'. $H, k \models K_i\phi$ iff for all $m \geq 0$, for all $H' \in \mathcal{H}$ such that $H_k \sim_i H'_m$, $H'_m, m \models \phi$
- K_i is an **S5** operator for each i .
 - A sound and complete axiomatization for knowledge and time can be found in [HvdMV, 2003] (linear time), [vdMW, 2003] (branching time) using a slightly different framework.

Background: Topologic

Topologic is a bimodal logic with a knowledge modality (K) and an effort modality (\diamond) first studied by Moss and Parikh in 1992

Background: Topologic

Topologic is a bimodal logic with a knowledge modality (K) and an effort modality (\diamond) first studied by Moss and Parikh in 1992

$$\phi \rightarrow \diamond K\phi$$

Background: Topologic

Topologic is a bimodal logic with a knowledge modality (K) and an effort modality (\diamond) first studied by Moss and Parikh in 1992

$$\phi \rightarrow \diamond K\phi$$

“if ϕ is true then after some ‘effort’ ϕ is known”

Topologic: Basic Model (one agent)

Subset Frame: $\langle W, \mathcal{O} \rangle$

- W is a set of states
- $\mathcal{O} \subseteq 2^W$ is a set of subsets of W , i.e., a set of *observations*

Neighborhood Situation: Given a subset frame $\langle W, \mathcal{O} \rangle$, (w, U) is called a neighborhood situation, where $w \in U$ and $U \in \mathcal{O}$.

Model: $\langle W, \mathcal{O}, V \rangle$, where $V : \Phi_0 \rightarrow 2^W$ is a valuation function.

Topologic: Truth

Truth: Formulas are interpreted at neighborhood situations.

- $w, U \models p$ iff $w \in V(p)$
- $w, U \models \neg\phi$ iff $w, U \not\models \phi$
- $w, U \models \phi \wedge \psi$ iff $w, U \models \phi$ and $w, U \models \psi$
- $w, U \models K\phi$ iff for all $v \in U$, $v, U \models \phi$
- $w, U \models \Diamond\phi$ iff there is a $V \in \mathcal{O}$ such that $w \in V \subseteq U$ and $w, V \models \phi$

Topologic: Core Axioms

1. All propositional tautologies, *MP*
2. $(p \rightarrow \Box p) \wedge (\neg p \rightarrow \Box \neg p)$, for $p \in \Phi_0$.
3. \Box is **S4**
4. K is **S5**
5. $K\Box\phi \rightarrow \Box K\phi$
6. K -NEC, \Box -NEC

Topologic: Some Results

- (Moss and Parikh: 1992) completely axiomatized Topologic for arbitrary subset spaces. (Dabrowski, Moss and Parikh: 1996)
- (Georgatos: 1993, 1994, 1997) completely axiomatized Topologic where \mathcal{O} is restricted to a topology and showed that the logic has the finite model property. Similarly for tree-like spaces.
- (Weiss and Parikh: 2002) showed that an infinite number of axiom schemes is required to axiomatize Topologics in which \mathcal{O} is closed under intersection.
- (Heinemann: 1999, 2001, 2003, 2004) has a number of papers in which temporal operators are added to the language. He also worked on Hybrid versions of Topologic

Topologic: Some Results

- (Moss and Parikh: 1992) completely axiomatized Topologic for arbitrary subset spaces. (Dabrowski, Moss and Parikh: 1996)
- (Georgatos: 1993, 1994, 1997) completely axiomatized Topologic where \mathcal{O} is restricted to a topology and showed that the logic has the finite model property. Similarly for tree-like spaces.
- (Weiss and Parikh: 2002) showed that an infinite number of axiom schemes is required to axiomatize Topologics in which \mathcal{O} is closed under intersection.
- (Heinemann: 1999, 2001, 2003, 2004) has a number of papers in which temporal operators are added to the language. He also worked on Hybrid versions of Topologic

Topologic: Some Results

- (Moss and Parikh: 1992) completely axiomatized Topologic for arbitrary subset spaces. (Dabrowski, Moss and Parikh: 1996)
- (Georgatos: 1993, 1994, 1997) completely axiomatized Topologic where \mathcal{O} is restricted to a topology and showed that the logic has the finite model property. Similarly for tree-like spaces.
- (Weiss and Parikh: 2002) showed that an infinite number of axiom schemes is required to axiomatize Topologics in which \mathcal{O} is closed under intersection.
- (Heinemann: 1999, 2001, 2003, 2004) has a number of papers in which temporal operators are added to the language. He also worked on Hybrid versions of Topologic

Topologic: Some Results

- (Moss and Parikh: 1992) completely axiomatized Topologic for arbitrary subset spaces. (Dabrowski, Moss and Parikh: 1996)
- (Georgatos: 1993, 1994, 1997) completely axiomatized Topologic where \mathcal{O} is restricted to a topology and showed that the logic has the finite model property. Similarly for tree-like spaces.
- (Weiss and Parikh: 2002) showed that an infinite number of axiom schemes is required to axiomatize Topologics in which \mathcal{O} is closed under intersection.
- (Heinemann: 1999, 2001, 2003, 2004) has a number of papers in which temporal operators are added to the language. He also worked on Hybrid versions of Topologic

First Issue

Choices an agent makes, at least partially, depends on the agents' states of knowledge.

In particular, when formalizing deontic situations, the role of knowledge is particularly important.

Certainly an agent cannot be faulted for not performing an action whose need it did not know about.

Some Examples

- a) Jill is a physician whose neighbour is ill. Jill does not know and has not been informed. Jill has no obligation (as yet) to treat the neighbour.
- b) Jill is a physician whose neighbour Sam is ill. The neighbour's daughter Ann comes to Jill's house and tells her. Now Jill does have an obligation to treat Sam, or perhaps call in an ambulance or a specialist.
- c) Mary is a patient in St. Gibson's hospital. Mary is having a heart attack. The caveat which applied in case a) does not apply here. The hospital has an obligation to be aware of Mary's condition at all times and to provide emergency treatment as appropriate.

Some Examples

- a) Jill is a physician whose neighbour is ill. Jill does not know and has not been informed. Jill has no obligation (as yet) to treat the neighbour.
- b) Jill is a physician whose neighbour Sam is ill. The neighbour's daughter Ann comes to Jill's house and tells her. Now Jill does have an obligation to treat Sam, or perhaps call in an ambulance or a specialist.
- c) Mary is a patient in St. Gibson's hospital. Mary is having a heart attack. The caveat which applied in case a) does not apply here. The hospital has an obligation to be aware of Mary's condition at all times and to provide emergency treatment as appropriate.

Some Examples

- a) Jill is a physician whose neighbour is ill. Jill does not know and has not been informed. Jill has no obligation (as yet) to treat the neighbour.
- b) Jill is a physician whose neighbour Sam is ill. The neighbour's daughter Ann comes to Jill's house and tells her. Now Jill does have an obligation to treat Sam, or perhaps call in an ambulance or a specialist.
- c) Mary is a patient in St. Gibson's hospital. Mary is having a heart attack. The caveat which applied in case a) does not apply here. The hospital has an obligation to be aware of Mary's condition at all times and to provide emergency treatment as appropriate.

d) Jill has a patient with a certain condition C who is in the St. Gibson hospital mentioned above. There are two drugs d and d' which can be used for C , but d has a better track record. Jill is about to inject the patient with d , but unknown to Jill, the patient is allergic to d and for this patient d' should be used. Nurse Rebecca is aware of the patient's allergy and also that Jill is about to administer d . It is then Rebecca's obligation to inform Jill and to suggest that drug d' be used in this case.

Actions

Assume a finite set, Act , of (*primitive*) actions that is a subset of

E . Given a finite global history H ,

$$a(H) = \{H' \mid Ha \preceq H' \text{ and } H' \text{ a global history}\}$$

Actions

Assume a finite set, Act , of (*primitive*) actions that is a subset of

E . Given a finite global history H ,

$$a(H) = \{H' \mid Ha \preceq H' \text{ and } H' \text{ a global history}\}$$

1. When an action is performed, it is performed at the next moment of time.
2. Only one agent can perform some action at any moment.
3. If no agents perform an action, then nature performs a ‘clock tick’.
4. Each agent knows *when* it can perform an action.

Actions

Assume a finite set, Act , of (*primitive*) actions that is a subset of E . Given a finite global history H ,

$$a(H) = \{H' \mid Ha \preceq H' \text{ and } H' \text{ a global history}\}$$

1. When an action is performed, it is performed at the next moment of time.
2. Only one agent can perform some action at any moment.
3. If no agents perform an action, then nature performs a ‘clock tick’. (synchronous systems)
4. Each agent knows *when* it can perform an action. ($\langle a \rangle \top \rightarrow K_i \langle a \rangle \top$)

Values: Informal Definition

All global histories will be presumed to have a **value**

Let $\mathcal{V}(H)$ be the set of extensions of (finite history) H which have the highest possible value. (Assumptions are needed to make $\mathcal{V}(H)$ well defined)

Values: Informal Definition

All global histories will be presumed to have a **value**

Let $\mathcal{V}(H)$ be the set of extensions of (finite history) H which have the highest possible value. (Assumptions are needed to make $\mathcal{V}(H)$ well defined)

We will say that a **is good** to be performed at H , $\mathcal{G}(a, H)$, if $\mathcal{V}(H) \subseteq a(H)$, i.e., there are no H -good histories which do not involve the performing of a .

And we say that a **may** be performed at H if $\mathcal{V}(H) \cap a(H)$ is non-empty.

Values: Formal Definition

Let \mathcal{H} be a set of global histories and $H \in \mathcal{H}$ a global history. For each $t \in \mathbb{N}$, let $\mathcal{F}(H_t) = \{H' \in \mathcal{H} \mid H_t \preceq H'\}$.

We require for each global history $H \in \mathcal{H}$,

1. For all $t \in \mathbb{N}$, $val[\mathcal{F}(H_t)]$ is a closed and bounded subset of \mathbb{R} .
2. $\bigcap_{t \in \mathbb{N}} val[\mathcal{F}(H_t)] = \{val(H)\}$

Define $\mathcal{V}(H_t) = \{H' \mid H' \in \operatorname{argmax}(val[\mathcal{F}(H_t)])\}$.

Thus $\mathcal{V}(H_t)$ is the set of maximally good, (or just maximal) extensions of H_t .

Knowledge Based Obligation

Agent i is obliged to perform action a at global history H and time t iff a is an action which i (only) can perform, and i *knows* that it is good to perform a .

For each $a \in Act$, we define a primitive proposition $G(a)$. We say that $H, t \models G(a)$ iff all maximal global histories H such that $H_t a \preceq H$.

Then we say that i is obliged to perform action a (at H, t) if $K_i(G(a))$ is true (at H, t).

Knowledge Based Obligation: Conclusions

- The formal framework just sketched can formalize the first three examples (a,b and c).
- Adapt the completeness proof of [HvdMV] to find a sound and complete axiomatization of logic of knowledge based obligation.
- Add **PDL** style calculus of actions ([vdM]).
- Notice that in the knowledge based obligation framework, acquiring knowledge may create an obligation, but it cannot erase (a persistent) one. Introduce the notion of **weak knowledge, or justified belief**.
- Is it possible to program the agents in such a way that *if the agents do what they know they ought to do*, then one of the best histories is produced?

Knowledge Based Obligation: Conclusions

- The formal framework just sketched can formalize the first three examples (a,b and c).
- Adapt the completeness proof of [HvdMV] to find a sound and complete axiomatization of logic of knowledge based obligation.
- Add **PDL** style calculus of actions ([vdM]).
- Notice that in the knowledge based obligation framework, acquiring knowledge may create an obligation, but it cannot erase (a persistent) one. Introduce the notion of **weak knowledge**, or **justified belief**.
- Is it possible to program the agents in such a way that *if the agents do what they know they ought to do*, then one of the best histories is produced?

Knowledge Based Obligation: Conclusions

- The formal framework just sketched can formalize the first three examples (a,b and c).
- Adapt the completeness proof of [HvdMV] to find a sound and complete axiomatization of logic of knowledge based obligation.
- Add **PDL** style calculus of actions ([vdM]).
- Notice that in the knowledge based obligation framework, acquiring knowledge may create an obligation, but it cannot erase (a persistent) one. Introduce the notion of **weak knowledge**, or **justified belief**.
- Is it possible to program the agents in such a way that *if the agents do what they know they ought to do*, then one of the best histories is produced?

Knowledge Based Obligation: Conclusions

- The formal framework just sketched can formalize the first three examples (a,b and c).
- Adapt the completeness proof of [HvdMV] to find a sound and complete axiomatization of logic of knowledge based obligation.
- Add **PDL** style calculus of actions ([vdM]).
- Notice that in the knowledge based obligation framework, acquiring knowledge may create an obligation, but it cannot erase (a persistent) one. Introduce the notion of **weak knowledge, or justified belief**.
- Is it possible to program the agents in such a way that *if the agents do what they know they ought to do*, then one of the best histories is produced?

Knowledge Based Obligation: Conclusions

- The formal framework just sketched can formalize the first three examples (a,b and c).
- Adapt the completeness proof of [HvdMV] to find a sound and complete axiomatization of logic of knowledge based obligation.
- Add **PDL** style calculus of actions ([vdM]).
- Notice that in the knowledge based obligation framework, acquiring knowledge may create an obligation, but it cannot erase (a persistent) one. Introduce the notion of **weak knowledge, or justified belief**.
- Is it possible to program the agents in such a way that *if the agents do what they know they ought to do*, then one of the best histories is produced?

Second Issue

The occurrence of an event, such as communication, changes the agents' states of knowledge. How should we model this *change* of information?

There has been a lot of recent papers about adding a notion of updating to otherwise static Kripke structures (Dynamic Epistemic Semantics).

Second Issue

The occurrence of an event, such as communication, changes the agents' states of knowledge. How should we model this *change* of information?

There has been a lot of recent papers about adding a notion of updating to otherwise static Kripke structures (Dynamic Epistemic Semantics).

In [PP]:

- Agents are assumed to communicate (only) according to a *communication graph*
- The starting point is not Kripke structures, but rather subset models (Topologic models).

The Logic of Communication Graphs: Introduction

- *Topologic*: a bimodal logic with a knowledge modality (K) and an effort modality (\diamond).

$$\phi \rightarrow \diamond K \phi$$

“if ϕ is true then after some ‘effort’ ϕ is known”

The Logic of Communication Graphs: Introduction

- *Topologic*: a bimodal logic with a knowledge modality (K) and an effort modality (\diamond).

$$\phi \rightarrow \diamond K \phi$$

“if ϕ is true then after some ‘effort’ ϕ is known”

- Extend to a multi-agent epistemic logic where **effort = communication**

The Logic of Communication Graphs: Introduction

- *Topologic*: a bimodal logic with a knowledge modality (K) and an effort modality (\diamond).

$$\phi \rightarrow \diamond K \phi$$

“if ϕ is true then after some ‘effort’ ϕ is known”

- Extend to a multi-agent epistemic logic where **effort = communication**

$$K_i \phi \rightarrow \diamond K_j \phi$$

The Logic of Communication Graphs: Introduction

- *Topologic*: a bimodal logic with a knowledge modality (K) and an effort modality (\diamond).

$$\phi \rightarrow \diamond K \phi$$

“if ϕ is true then after some ‘effort’ ϕ is known”

- Extend to a multi-agent epistemic logic where **effort = communication**

$$K_i \phi \rightarrow \diamond K_j \phi$$

“if i knows ϕ then after some communication j knows ϕ ”

An Example

If Bush wants some information from a particular CIA operative, say Bob, he must get this information through Tenet. Suppose that ϕ is the exact whereabouts of Bin Laden and

$$K_{\text{Bob}}\phi \wedge \neg K_{\text{Bush}}\phi$$

An Example

If Bush wants some information from a particular CIA operative, say Bob, he must get this information through Tenet. Suppose that ϕ is the exact whereabouts of Bin Laden and

$$K_{\text{Bob}}\phi \wedge \neg K_{\text{Bush}}\phi$$

Bush *can* find out the exact whereabouts of Bin Laden

An Example

If Bush wants some information from a particular CIA operative, say Bob, he must get this information through Tenet. Suppose that ϕ is the exact whereabouts of Bin Laden and

$$K_{\text{Bob}}\phi \wedge \neg K_{\text{Bush}}\phi$$

Bush *can* find out the exact whereabouts of Bin Laden

$$\diamond K_{\text{Bush}}\phi$$

An Example

If Bush wants some information from a particular CIA operative, say Bob, he must get this information through Tenet. Suppose that ϕ is the exact whereabouts of Bin Laden and

$$K_{\text{Bob}}\phi \wedge \neg K_{\text{Bush}}\phi$$

Bush *can* find out the exact whereabouts of Bin Laden

$$\diamond K_{\text{Bush}}\phi$$

but of course, *we* cannot find out such information

An Example

If Bush wants some information from a particular CIA operative, say Bob, he must get this information through Tenet. Suppose that ϕ is the exact whereabouts of Bin Laden and

$$K_{\text{Bob}}\phi \wedge \neg K_{\text{Bush}}\phi$$

Bush *can* find out the exact whereabouts of Bin Laden

$$\diamond K_{\text{Bush}}\phi$$

but of course, *we* cannot find out such information

$$\bigwedge_{i \in 4421} \neg \diamond K_i \phi$$

A *pre-requisite* for Bush learning ϕ , Tenet will also have come to know ϕ .

A *pre-requisite* for Bush learning ϕ , Tenet will also have come to know ϕ .

$$\neg K_{\text{Bush}} \phi \wedge \Box (K_{\text{Bush}} \phi \rightarrow K_{\text{Tenet}} \phi)$$

A *pre-requisite* for Bush learning ϕ , Tenet will also have come to know ϕ .

$$\neg K_{\text{Bush}} \phi \wedge \Box(K_{\text{Bush}} \phi \rightarrow K_{\text{Tenet}} \phi)$$

So, there is a connection between the fact that the CIA restricts the flow of information and the truth of certain formulas in our language, such as

$$\bigwedge_{i \in 4421} \neg \Diamond K_i \phi \quad \text{and} \quad \neg K_{\text{Bush}} \phi \wedge \Box(K_{\text{Bush}} \phi \rightarrow K_{\text{Tenet}} \phi)$$

Communication Graphs

Let A be a set of agents. A **communication graph** is a directed graph $G_A = (A, E)$ such that for all $i \in A$, $(i, i) \notin E$

$(i, j) \in E$ means that i can directly receive information from agent j , *without* j knowing this fact.

Assume that:

1. all the agents share a common language
2. the agents make available all possible pieces of information

Uncertainty of the agents

1. Agents are uncertain about the actual state of the world.

Uncertainty of the agents

1. Agents are uncertain about the actual state of the world.
2. Agents are uncertain about the structure used to represent the information possessed by the agents, i.e., uncertainty about the Kripke structure.

Uncertainty of the agents

1. Agents are uncertain about the actual state of the world.
2. Agents are uncertain about the structure used to represent the information possessed by the agents, i.e., uncertainty about the Kripke structure.
 - (a) Agents are uncertain about the communication among agents not reachable via the communication graph.

Uncertainty of the agents

1. Agents are uncertain about the actual state of the world.
2. Agents are uncertain about the structure used to represent the information possessed by the agents, i.e., uncertainty about the Kripke structure.
 - (a) Agents are uncertain about the communication among agents not reachable via the communication graph.
 - (b) Suppose that agent i learns some information P from agent j 's website. Agent i is uncertain as to *how* agent j came to know P , i.e., what questions did j ask and to whom.

A Logic for Communication Graphs: Semantics

Fix a communication graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$ and a set of states W .

A Logic for Communication Graphs: Semantics

Fix a communication graph $\mathcal{G} = (\mathcal{A}, E)$ and a set of states W .

A **communication event** is a tuple (ϕ, i, j) intended to mean that i learns information ϕ from j , where

- ϕ is a ground (propositional) formula
- there is an edge between i and j in \mathcal{G}

A Logic for Communication Graphs: Semantics

Fix a communication graph $\mathcal{G} = (\mathcal{A}, E)$ and a set of states W .

A **communication event** is a tuple (ϕ, i, j) intended to mean that i learns information ϕ from j , where

- ϕ is a ground (propositional) formula
- there is an edge between i and j in \mathcal{G}

Let $\vec{X} = (X_1, \dots, X_n)$ be a vector of (non-empty) sets of states – intuitively, the states that each agent consider possible.

A Logic for Communication Graphs: Semantics

Fix a communication graph $\mathcal{G} = (\mathcal{A}, E)$ and a set of states W .

A **communication event** is a tuple (ϕ, i, j) intended to mean that i learns information ϕ from j , where

- ϕ is a ground (propositional) formula
- there is an edge between i and j in \mathcal{G}

Let $\vec{X} = (X_1, \dots, X_n)$ be a vector of (non-empty) sets of states – intuitively, the states that each agent consider possible.

An **event** is of two types:

1. an **initial event** is a vector \vec{X} , or

A Logic for Communication Graphs: Semantics

Fix a communication graph $\mathcal{G} = (\mathcal{A}, E)$ and a set of states W .

A **communication event** is a tuple (ϕ, i, j) intended to mean that i learns information ϕ from j , where

- ϕ is a ground (propositional) formula
- there is an edge between i and j in \mathcal{G}

Let $\vec{X} = (X_1, \dots, X_n)$ be a vector of (non-empty) sets of states – intuitively, the states that each agent consider possible.

An **event** is of two types:

1. an **initial event** is a vector \vec{X} , or
2. a pair $((\phi, i, j), \vec{X})$, where (ϕ, i, j) is a communication event.

A Logic for Communication Graphs: Semantics

Let Σ_g denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_g^*$.

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

1. If $H = e$ where e is an initial event, then H is consistent

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

1. If $H = e$ where e is an initial event, then H is consistent
2. Given a consistent history H , where \vec{X} is the vector associated with the last event in H and $e = ((\phi, i, j), \vec{Y})$, then He is consistent provided

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

1. If $H = e$ where e is an initial event, then H is consistent
2. Given a consistent history H , where \vec{X} is the vector associated with the last event in H and $e = ((\phi, i, j), \vec{Y})$, then He is consistent provided
 - i learns ϕ ($X_i = Y_i \cap \hat{\phi}$)

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

1. If $H = e$ where e is an initial event, then H is consistent
2. Given a consistent history H , where \vec{X} is the vector associated with the last event in H and $e = ((\phi, i, j), \vec{Y})$, then He is consistent provided
 - i learns ϕ ($X_i = Y_i \cap \hat{\phi}$)
 - j knows ϕ ($X_j \subseteq \hat{\phi}$)

A Logic for Communication Graphs: Semantics

Let Σ_G denote the set of all events.

A **history** is any finite sequence of events, i.e., $H \in \Sigma_G^*$.

Let \mathcal{H}_G be the set of all **consistent** histories, where ‘consistent’ is defined recursively as follows:

1. If $H = e$ where e is an initial event, then H is consistent
2. Given a consistent history H , where \vec{X} is the vector associated with the last event in H and $e = ((\phi, i, j), \vec{Y})$, then He is consistent provided
 - i learns ϕ ($X_i = Y_i \cap \hat{\phi}$)
 - j knows ϕ ($X_j \subseteq \hat{\phi}$)
 - $p \neq i$ is unaware of the communication ($Y_p = X_p$, for $p \neq i$)

Some notation: Let $H \in \mathcal{H}_G$, define

- H^0 is the initial event of H (a vector of sets)
- $\text{Comm}(H)$ is the sequence of communication events generated by H
- H^k is the vector of sets associated with the second component of the k th event of the sequence
- So, H_i^k is the set of states the i considers possible at moment k

Given a sequence K of communication events let $\lambda_i(K)$ be the subsequence of communication events seen by agent i (i is in the second component)

The Logic of Communication Graphs: Truth

Given two events $H, F \in \mathcal{H}_G$, we say $H \sim_i F$ iff $H^0 = F^0$ and

$$\lambda_i(\text{Comm}(H)) = \lambda_i(\text{Comm}(F))$$

The Logic of Communication Graphs: Truth

Given two events $H, F \in \mathcal{H}_G$, we say $H \sim_i F$ iff $H^0 = F^0$ and $\lambda_i(\text{Comm}(H)) = \lambda_i(\text{Comm}(F))$

Truth is defined at pairs (w, H) where $w \in W$ and $H \in \mathcal{H}_G$.

The Logic of Communication Graphs: Truth

Given two events $H, F \in \mathcal{H}_G$, we say $H \sim_i F$ iff $H^0 = F^0$ and $\lambda_i(\text{Comm}(H)) = \lambda_i(\text{Comm}(F))$

Truth is defined at pairs (w, H) where $w \in W$ and $H \in \mathcal{H}_G$.

- $w, H \models \diamond\phi$ iff $\exists H', H \preceq H'$ and $w, H' \models \phi$

The Logic of Communication Graphs: Truth

Given two events $H, F \in \mathcal{H}_G$, we say $H \sim_i F$ iff $H^0 = F^0$ and $\lambda_i(\text{Comm}(H)) = \lambda_i(\text{Comm}(F))$

Truth is defined at pairs (w, H) where $w \in W$ and $H \in \mathcal{H}_G$.

- $w, H \models \diamond\phi$ iff $\exists H', H \preceq H'$ and $w, H' \models \phi$
- $w, H \models K_i\phi$ iff for all $H' \sim_i H$, $w, H' \models \phi$

Topologic: Conclusions

- Core Topologic axioms are sound. (Complete?)
- Connections between valid formulas and the communication graph.

$$K_j\phi \wedge \neg K_l\phi \rightarrow \diamond(K_i\phi \wedge \neg K_l\phi)$$

- Decidability of the logic of communication graphs.
- Comparison with dynamic epistemic semantics ($[\alpha]\phi$, where α is an epistemic program)
- Relax the strong assumptions we are making about the type of communication that can take place between agents.

Topologic: Conclusions

- Core Topologic axioms are sound. (Complete?)
- Connections between valid formulas and the communication graph.

$$K_j\phi \wedge \neg K_l\phi \rightarrow \diamond(K_i\phi \wedge \neg K_l\phi)$$

- Decidability of the logic of communication graphs.
- Comparison with dynamic epistemic semantics ($[\alpha]\phi$, where α is an epistemic program)
- Relax the strong assumptions we are making about the type of communication that can take place between agents.

Topologic: Conclusions

- Core Topologic axioms are sound. (Complete?)
- Connections between valid formulas and the communication graph.

$$K_j\phi \wedge \neg K_l\phi \rightarrow \diamond(K_i\phi \wedge \neg K_l\phi)$$

- Decidability of the logic of communication graphs.
- Comparison with dynamic epistemic semantics ($[\alpha]\phi$, where α is an epistemic program)
- Relax the strong assumptions we are making about the type of communication that can take place between agents.

Topologic: Conclusions

- Core Topologic axioms are sound. (Complete?)
- Connections between valid formulas and the communication graph.

$$K_j\phi \wedge \neg K_l\phi \rightarrow \diamond(K_i\phi \wedge \neg K_l\phi)$$

- Decidability of the logic of communication graphs.
- Comparison with dynamic epistemic semantics ($[\alpha]\phi$, where α is an epistemic program)
- Relax the strong assumptions we are making about the type of communication that can take place between agents.

Topologic: Conclusions

- Core Topologic axioms are sound. (Complete?)
- Connections between valid formulas and the communication graph.

$$K_j\phi \wedge \neg K_l\phi \rightarrow \diamond(K_i\phi \wedge \neg K_l\phi)$$

- Decidability of the logic of communication graphs.
- Comparison with dynamic epistemic semantics ($[\alpha]\phi$, where α is an epistemic program)
- Relax the strong assumptions we are making about the type of communication that can take place between agents.

Third Issue

Whether made explicit or implicit, knowledge theoretic properties such as common knowledge of rationality are important in understanding and modelling game-theoretic situations.

Much of the literature is focused on what is assumed about agents' knowledge in order to make various solution concepts "effective".

We show that the agents need a certain amount of information in order for the Gibbard-Sattherwaite theorem to be "effective".

An Example

| Size | Group | I | II | III | IV |
|------|-------|-----------|-----------|-----------|-----------|
| 40 | A | o1 | <i>o1</i> | <i>o4</i> | o1 |
| 30 | B | <i>o2</i> | o2 | o2 | <i>o2</i> |
| 15 | C | <i>o3</i> | o2 | o2 | <i>o2</i> |
| 8 | D | <i>o4</i> | <i>o4</i> | <i>o1</i> | <i>o4</i> |
| 7 | E | <i>o3</i> | <i>o3</i> | <i>o1</i> | o1 |

$$P_A^* = (\mathbf{o}_1, o_4, o_2, o_3)$$

$$P_B^* = (o_2, o_1, o_3, o_4)$$

$$P_C^* = (o_3, o_2, o_4, o_1)$$

$$P_D^* = (o_4, o_1, o_2, o_3)$$

$$P_E^* = (o_3, o_1, o_2, o_4)$$

If the current winner is o , then agent i will switch its vote to some candidate o' provided

1. i prefers o' to o , and
2. the current total for o' plus agent i 's votes for o' is greater than the current total for o .

An Example

$$P_A^* = (o_1, o_4, o_2, o_3)$$

$$P_B^* = (o_2, o_1, o_3, o_4)$$

$$P_C^* = (o_3, o_2, o_4, o_1)$$

$$P_D^* = (o_4, o_1, o_2, o_3)$$

$$P_E^* = (o_3, o_1, o_2, o_4)$$

| Size | Group | I | II | III | IV |
|------|-------|-----------|-----------|-----------|-----------|
| 40 | A | o1 | o1 | o4 | o1 |
| 30 | B | o2 | o2 | o2 | o2 |
| 15 | C | o3 | o2 | o2 | o2 |
| 8 | D | o4 | o4 | o1 | o4 |
| 7 | E | o3 | o3 | o1 | o1 |

If the current winner is o , then agent i will switch its vote to some candidate o' provided

1. i prefers o' to o , and
2. the current total for o' plus agent i 's votes for o' is greater than the current total for o .

An Example

$$P_A^* = (o_1, o_4, o_2, o_3)$$

$$P_B^* = (o_2, o_1, o_3, o_4)$$

$$P_C^* = (o_3, o_2, o_4, o_1)$$

$$P_D^* = (o_4, o_1, o_2, o_3)$$

$$P_E^* = (o_3, o_1, o_2, o_4)$$

| Size | Group | I | II | III | IV |
|-----------|-------|-----------|-----------|------------|-----------|
| 40 | A | o1 | o1 | III | IV |
| 30 | B | <i>o2</i> | o2 | o2 | <i>o2</i> |
| 15 | C | <i>o3</i> | o2 | o2 | <i>o2</i> |
| 8 | D | <i>o4</i> | o4 | o1 | <i>o4</i> |
| 7 | E | <i>o3</i> | o3 | o1 | o1 |

If the current winner is o , then agent i will switch its vote to some candidate o' provided

1. i prefers o' to o , and
2. the current total for o' plus agent i 's votes for o' is greater than the current total for o .

An Example

| Size | Group | I | II | III | IV |
|------|-------|-----------|-----------|-----------|-----------|
| 40 | A | o1 | <i>o1</i> | <i>o4</i> | o1 |
| 30 | B | <i>o2</i> | o2 | o2 | <i>o2</i> |
| 15 | C | <i>o3</i> | o2 | o2 | <i>o2</i> |
| 8 | D | <i>o4</i> | <i>o4</i> | <i>o1</i> | <i>o4</i> |
| 7 | E | <i>o3</i> | <i>o3</i> | <i>o1</i> | o1 |

$$P_A^* = (o_1, o_4, o_2, o_3)$$

$$P_B^* = (o_2, o_1, o_3, o_4)$$

$$P_C^* = (o_3, o_2, o_4, o_1)$$

$$P_D^* = (o_4, o_1, o_2, o_3)$$

$$P_E^* = (o_3, o_1, o_2, o_4)$$

If the current winner is o , then agent i will switch its vote to some candidate o' provided

1. i prefers o' to o , and
2. the current total for o' plus agent i 's votes for o' is greater than the current total for o .

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i's true preference*
- Let \mathbf{Pref} denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i*'s true preference
- Let \mathbf{Pref} denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i 's true preference*
- Let \mathbf{Pref} denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i*'s true preference
- Let \mathbf{Pref} denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i*'s **true preference**
- Let **Pref** denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

Towards a Logic for Voting Models

Refer to [B] and [BF] for a complete discussion of formal models of voting systems.

- Let $\mathcal{O} = \{o_1, \dots, o_m\}$ be a set of candidates,
- $A = \{1, \dots, n\}$ be a set of agents or voters.
- We assume that each voter has a strict preference over the elements of \mathcal{O} , i.e., a reflexive, transitive and connected relation on \mathcal{O} . We represent each P_i by a function $P_i : \mathcal{O} \rightarrow \{1, \dots, m\}$
- Let P_i^* represent *i*'s true preference
- Let **Pref** denote the set of preferences over \mathcal{O} .
- A *preference profile* is an element of $(\mathbf{Pref})^n$.

An **aggregation function** is a function from preference profiles to preferences:

$$\mathbf{Ag} : \mathbf{Pref}^n \rightarrow \mathbf{Pref}$$

An **aggregation function** is a function from preference profiles to preferences:

$$\mathbf{Ag} : \mathbf{Pref}^n \rightarrow \mathbf{Pref}$$

Typically, agents select a vote that ‘represents’ their preference.

An **aggregation function** is a function from preference profiles to preferences:

$$\text{Ag} : \text{Pref}^n \rightarrow \text{Pref}$$

Typically, agents select a vote that ‘represents’ their preference.

There are two components to any voting procedure:

An **aggregation function** is a function from preference profiles to preferences:

$$\text{Ag} : \text{Pref}^n \rightarrow \text{Pref}$$

Typically, agents select a vote that ‘represents’ their preference.

There are two components to any voting procedure:

1. The type of votes that voters can cast. Let $\mathcal{B}(\mathcal{O})$ be the set of feasible votes, or *ballots*.

An **aggregation function** is a function from preference profiles to preferences:

$$\text{Ag} : \text{Pref}^n \rightarrow \text{Pref}$$

Typically, agents select a vote that ‘represents’ their preference.

There are two components to any voting procedure:

1. The type of votes that voters can cast. Let $\mathcal{B}(\mathcal{O})$ be the set of feasible votes, or *ballots*.
2. A tallying function $\text{Ag}_v : \mathcal{B}(\mathcal{O})^n \rightarrow \mathcal{O}$ maps vote profiles to candidates.

Strategizing

An element $\vec{v} \in \mathcal{B}(\mathcal{O})^n$ is called a **vote profile**.

Strategizing

An element $\vec{v} \in \mathcal{B}(\mathcal{O})^n$ is called a **vote profile**.

Fix a voting procedure. Let $\mathcal{B}(P) \subseteq \mathcal{B}(\mathcal{O})$ be the set of votes which faithfully represent preference P .

Strategizing

An element $\vec{v} \in \mathcal{B}(\mathcal{O})^n$ is called a **vote profile**.

Fix a voting procedure. Let $\mathcal{B}(P) \subseteq \mathcal{B}(\mathcal{O})$ be the set of votes which faithfully represent preference P .

The voter i is said to **strategize** if i selects a vote v that is not in the set $\mathcal{B}(P_i)$. (Eg. If using plurality voting, then $\mathcal{B}(P_i) = \{\operatorname{argmax}(P_i)\}$)

In what follows we assume that agent i always select an element of $\mathcal{B}(P_i)$, so **strategizing** will mean choosing an element of \mathbf{Pref} different from i 's true preference P_i^* .

Strategizing

An element $\vec{v} \in \mathcal{B}(\mathcal{O})^n$ is called a **vote profile**.

Fix a voting procedure. Let $\mathcal{B}(P) \subseteq \mathcal{B}(\mathcal{O})$ be the set of votes which faithfully represent preference P .

The voter i is said to **strategize** if i selects a vote v that is not in the set $\mathcal{B}(P_i)$. (Eg. If using plurality voting, then $\mathcal{B}(P_i) = \{\operatorname{argmax}(P_i)\}$)

In what follows we assume that agent i always select an element of $\mathcal{B}(P_i)$, so **strategizing** will mean choosing an element of **Pref** different from i 's true preference P_i^* .

Strategizing Functions

Given a profile $\vec{\mathcal{P}}$ of *actual* votes, we ask whether agent i will change its vote if given another chance to express its preference.

Strategizing Functions

Given a profile $\vec{\mathcal{P}}$ of *actual* votes, we ask whether agent i will change its vote if given another chance to express its preference.

Then given $\vec{\mathcal{P}}_{-i}$ and i 's true preference P_i^* , there will be a (nonempty) set X_i of those preferences that are i 's best response to $\vec{\mathcal{P}}_{-i}$.

Strategizing Functions

Given a profile $\vec{\mathcal{P}}$ of *actual* votes, we ask whether agent i will change its vote if given another chance to express its preference.

Then given $\vec{\mathcal{P}}_{-i}$ and i 's true preference P_i^* , there will be a (nonempty) set X_i of those preferences that are i 's best response to $\vec{\mathcal{P}}_{-i}$.

Suppose that $f_i(\vec{\mathcal{P}}_{-i})$ selects one such best response from X_i .

Strategizing Functions

Given a profile $\vec{\mathcal{P}}$ of *actual* votes, we ask whether agent i will change its vote if given another chance to express its preference.

Then given $\vec{\mathcal{P}}_{-i}$ and i 's true preference P_i^* , there will be a (nonempty) set X_i of those preferences that are i 's best response to $\vec{\mathcal{P}}_{-i}$.

Suppose that $f_i(\vec{\mathcal{P}}_{-i})$ selects one such best response from X_i .

Let $f(\vec{\mathcal{P}}) = (f_1(\vec{\mathcal{P}}_{-1}), \dots, f_n(\vec{\mathcal{P}}_{-n}))$. We call f a **strategizing function**.

Some Results

If $\vec{\mathcal{P}}$ is a fixed point of f (i.e., $f(\vec{\mathcal{P}}) = \vec{\mathcal{P}}$), then $\vec{\mathcal{P}}$ is a **stable** outcome.

Theorem 1 (CPP) *For any given tallying function Ag_v , there exists an initial vector of preferences such that f never stabilizes.*

Theorem 2 (CPP) *Fix a voting model $\langle A, \mathcal{O}, \{P_i^*\}_{i \in A}, \text{Ag}, f \rangle$ and a **knowledge graph** $\mathcal{K} = (A, E_{\mathcal{K}})$. If \mathcal{K} is directed and acyclic then the strategizing function f will stabilize at level k , where k is the height of the graph \mathcal{K} ; f can cycle only if the associated knowledge graph has a cycle.*

Conclusions and Further Work

- Theory of correctness of social procedures.
- Applications: A formal theory of correctness of social software could be used to determine whether the procedure under consideration is “sound” and “complete” for its intended interpretation.

Let P be a social procedure:

P is **sound** if by following the rules of the procedure no unintended consequences are generated

P is **complete** for social situation S if all intended consequences of S can be achieved by following the rules of P .

Conclusions and Further Work

- Theory of correctness of social procedures.
- Applications: A formal theory of correctness of social software could be used to determine whether the procedure under consideration is “sound” and “complete” for its intended interpretation.

Let P be a social procedure:

P is **sound** if by following the rules of the procedure no unintended consequences are generated

P is **complete** for social situation S if all intended consequences of S can be achieved by following the rules of P .

Conclusions and Further Work

- There are a wealth of social procedures described in the game theory literature. Attempts to formalize these procedures create a wide range of interesting problems for logicians. For example, see [PS] for a logic that axiomatizes the concept of majority. **It will be very interesting to see whether a formal analysis can help create new procedures or refine old ones.**
- Move to an explicit language (Artemov, Fitting): Replace $K_i\phi$ with formulas of the form $t:\phi$, intended to mean that t is a justification, or evidence, for ϕ . **It would be interesting to see what effect a shift to this explicit would have on the issues discussed in this paper.**

Conclusions and Further Work

- There are a wealth of social procedures described in the game theory literature. Attempts to formalize these procedures create a wide range of interesting problems for logicians. For example, see [PS] for a logic that axiomatizes the concept of majority. It will be very interesting to see whether a formal analysis can help create new procedures or refine old ones.
- Move to an explicit language (Artemov, Fitting): Replace $K_i\phi$ with formulas of the form $t:\phi$, intended to mean that t is a justification, or evidence, for ϕ . It would be interesting to see what effect a shift to this explicit would have on the issues discussed in this paper.

Thank you.