Knowledge and Structure in Social Algorithms

Rohit Parikh

rparikh@gc.cuny.edu

City University of New York

IIT Kanpur, January 2008

.

Abstract: *The first third of the XXth century saw two important developments. One of these was Ramsey's tracing of personal probabilities to an agent's choices. This was a precursor to the work of de Finetti, von Neumann and Morgenstern, and Savage. The other one was Turing's invention of the Turing machine and the formulation of the Church-Turing thesis according to which all computable functions on natural numbers were recursive or Turing computable.*

.

Game theory has depended heavily on the first of these developments, since of course von Neumann and Morgenstern can be regarded as the fathers of Game theory. But the other development has received less attention. This development led to the development and design of computers and also to fields like logic of programs, complexity theory and analysis of algorithms. It also resulted in much deeper understanding of algorithms, but only computer algorithms. Social algorithms remain largely unanalyzed except in special subfields like social choice theory or fair division [2]. These

fields do not tend to analyze complex social algorithms as is done in computer science.

.

A later development, going back to the work of Hintikka, Lewis and a little later Aumann [5, 6, 1], brought in the issue of knowledge. *The notion of common knowledge is of course very important for Aumann as* common knowledge of rationality *can be seen as a justification for backward induction arguments.*

But knowledge too has received less attention than it might. We all know that the Plame affair had something to do with someone knowing something which they should not have, and someone revealing something which they should not have. But why should they not? Clearly because of certain possible consequences. Knowledge and knowledge transfer are ubiquitous in how social algorithms work.

.

We will try in this talk to bring attention to the importance of the two issues of knowledge and logical structure of algorithms, and show the way to a broader arena in which game theorists might want to play. Hopefully, in fact almost certainly, there is a rich general theory to be developed.

.

1 Introduction

The notion of algorithm is implicit in so many things which happen in everyday life. We humans are tool-making creatures (as are chimps to a somewhat smaller extent) and both individual and social life is over-run with routines, from cooking recipes (Savage's celebrated eggs to omelette example) to elec-

tions – a subject of much discussion going back to Condorcet.

Over the last ten years or so, a field called **Social Software** [11] has come into existence which carries out a systematic study of such questions, and the purpose of this paper is to give an introduction to the issues. We will proceed by means of examples..

2 Structure

Normally, a piece of social software or social algorithm has a logical structure. As was argued in [11], this structure must address three important aspects, namely incentives, knowledge, and logical structure. For normally, an algorithm has logical structure, "A happens before B, which is succeeded by action C if condition X holds and by action D if X does not hold."

But quite often, the logical structure of the algorithm is parasitic on logical (or algorithmic) properties of existing physical or social structures. Clearly a prison needs a certain physical structure in order to be able to confine people, and a classroom needs a blackboard or a lectern in order for it to be usable as the venue of a lecture. Thus the teacher can now perform actions like write "No class tomorrow" on the blackboard and the students can read what she wrote or copy it in their notebooks. The physical properties of the blackboard enable certain actions with their own algorithmic properties..

A *social structure* with certain logical properties is a queue.

The queue is a very popular institution which occurs both in daily life and in computer programs. In a computer program, a queue is a FIFO structure, where FIFO means, "First in, first out." There are two operations, one by which an element is deleted from the front of the queue, and a second one where an element is added to the back of the queue. In real life, the queue could consist of people waiting at a bank to be served. The person deleted is the one who was at the front of the queue but is no longer in the queue,

and who receives service from a teller. An element which is added is a new customer who has just arrived and who goes to the back of the queue..

Clearly the queue implements our notions of fairness, (which can be proved rigorously as a theorem) that someone who came earlier gets service earlier, and in a bank this typically does happen. If someone in a bank tries to rush to the head of the line, people will stop him. We also have queues at bus stops and quite often the queue breaks down, there is a rush for seats at the last moment. Presumably the difference arises because things happen much faster in a bus queue than they do in a bank. At a bus stop, when the bus arrives, everything happens very fast and people are more interested in getting *on* the bus than in enforcing the rules..

Consider now, by comparison, the problem of parking, which is a similar problem. A scarce resource needs to be allocated on the basis of some sort of priority, which, however, is difficult to determine. When people are looking for parking in a busy area, they tend to cruise around until they find a space. There is no queue as such, but in general we do want that someone who arrives first should find a parking space and someone who arrives later may not. This is much more likely in a university or company parking lot, which is compact, and may even have a guard, rather than on the street, where parking is distributed, and priority does play *some* role but it is a probabilistic role. This phenomenon has unfortunate consequences as Shoup [17] points out..

> *When my students and I studied cruising for parking in a 15-block business district in Los Angeles, we found the average cruising time was 3.3 minutes, and the average cruising distance half a mile (about 2.5 times around the block). This may not sound like much, but with 470 parking meters in the district, and a turnover rate for curb parking of 17 cars per space per day, 8,000 cars park at the*

curb each weekday. Even a small amount of cruising time for each car adds up to a lot of traffic..

Over the course of a year, the search for curb parking in this 15-block district created about 950,000 excess vehicle miles of travel – equivalent to 38 trips around the earth, or four trips to the moon. And here's another inconvenient truth about underpriced curb parking: cruising those 950,000 miles wastes 47,000 gallons of gas and produces 730 tons of the greenhouse gas carbon dioxide. If all this happens in one small business district, imagine the cumulative effect of all cruising in the United States..

Shoup regards this problem as one of incentive and suggests that parking fees be raised so that occupancy of street parking spaces is only 85%. But clearly this will penalize the less affluent drivers. The new fees will likely be still less than garage parking, affluent drivers will abandon garage parking for street parking, and the less affluent drivers will be priced out. Note by contrast that we do not need to charge people for standing in a queue. Surely queues would also be shorter if people had to pay to stand in them, but this has not occurred to anyone as a solution to the 'standing in line problem.'.

Ultimately, the difference between queues and searching for parking is structural. In one case there is an easy algorithmic solution which respects priority (more or less) and in the other case such solutions are harder to find – except when we are dealing with parking lots.

An algorithmic solution might well be possible using something like a GPS system. If information about empty parking spaces was available to a central computer which could also accept requests from cars for parking spaces, and allocate spaces to arriving cars, then a solution could in fact be implemented. The information transfer and the allocation system would in effect convert the physically distributed parking spaces into the algorithmic equivalent of a

queue. .

Here is another example. When you rent an apartment, you receive a key from the landlord. The key serves two purposes. Its possession is *proof of a right*, the right to enter the apartment. But its possession is also a *physical enabler*. The two are not the same of course, since if you lose your key, you still have the right but are not enabled. If some stranger finds the key, then he is enabled, but does not have the right. Thus the two properties of a key do not coincide perfectly. But normally the two do coincide..

There are other analogs of a key which perform similar functions to a key. A password to a computer account is like a key, but does not need to be carried in your pocket. An ID card establishes your right to enter, but you typically a guard to be present and to see your card and let you into the building. If the building is locked and the guard is not present, you are out of luck.

In any case, these various generalized keys differ in some crucial ways. Stealing someone's identity was at one time very difficult. You had to look like that person, know some personal facts, and you had to stay away from that person's dog who knew perfectly well that you had the wrong smell. You needed a different 'ID' for the dog than you needed for people..

But nowadays identity theft is extremely easy. Lots of Social Security numbers, and mothers' maiden names are out there for the taking, and people who do not look like you at all can make use of them. Personal appearance or brass keys which originally provided proof of "right to entry," have been replaced by electronic items which are very easy to steal..

Let x be an individual, and let $R(x)$ mean that x has a right to use the key and $E(x)$ mean that x is enabled by the key. Then we have two important

conditions.

- **Safety:** $E(x) \rightarrow R(x)$. Whoever is enabled has the right
- **Liveness:** $R(x) \rightarrow E(x)$. Whoever has the right is enabled.

Of course safety could be thought of in terms of the contrapositive, $\sim R(x) \rightarrow \sim E(X)$, whoever does not have the right is not enabled. Clearly, safety is more important than liveness. If you lose your key and someone finds it, you are in trouble. But liveness also matters. A good notion of *key* must provide for both properties..

In any case the structural problem (of safety) can be addressed at the incentive level, for instance by instituting heavy penalties for stealing identities. But we could also look for a structural solution without seeking to penalize anyone.

Toddlers are apt to run away and into trouble, but we do not solve the problem by punishing them – we solve it by creating barriers to such escape, e.g., safety gates..

Another interesting example is a fence. A farmer may have a fence to keep his sheep in, and the fence prevents certain kinds of movement – namely sheep running away. But sometimes on a university campus we will see a very low fence around a grassy area. Almost anyone can walk over the fence, so the fence is no longer a physical obstacle. Rather the value of the fence is now informational, it says, *Thou shalt not cross!* With the yellow tape which the police sometimes put up, perhaps around a crime scene, or perhaps simply to block off some intersection, the *Thou shalt not cross* acquires quite a bit of punch..

3 Knowledge

Distributed Algorithms are much studied by computer scientists. A lot of commercial activity which goes on on the web has the property of being a distributed algorithm with many players. And of course the market is itself a very old distributed algorithm.

In such algorithms, it is crucial to make sure that when agents have to act, they have the requisite knowledge. And models for calculating such knowledge have existed for some time; we ourselves have participated in constructing such models [16, 15]. See also [3].

.

The notion of *common knowledge* as the route to consensus was introduced by Aumann in [1]. There is subsequent work by various people, including Geanakoplos and Polemarchakis [4] and ourselves [13]. Aumann simply assumed common knowledge, and showed that two agents would agree on the value of a random variable if they had common knowledge of their beliefs about it. [4] showed that even if the agents did not have common knowledge to start with, if they exchanged values, they would arrive at consensus, and common knowledge of that fact. Parikh and Krasucki [13] carried this one step further and considered many agents exchanging values in *pairwise interactions*. No common knowledge could now arise, as most agents would remain unaware of individual transactions they were not a party to. Nonetheless there would be consensus. Thus this exchange of values could be seen as a distributed algorithm which achieved a result.

.

A brief overview of the [PK] result:

Suppose we have n agents connected in a strongly connected graph. They all share initial probability distribution, but have now received, each of them, a finite amount of private information. Thus their estimate of the probability

of some event or the expected value of some random variable v may now be different.

Let g be a function which, at stage n picks out a sender $s(n)$ and a recipient $r(n)$. $s(n)$ sends his latest value of v to $r(n)$ who then revises *her* valuation of v.

If the graph G is strongly connected, and for each pair of connected agents i, j, i repeatedly sends his value of v to j, then eventually all estimates of the value of v become equal.

.

Issues about how knowledge enters into social algorithms are discussed in [7, 9, 16].

[16] actually ddiscusses how a framework for defining knowledge can be developed. A finite number of agents have some private information to start with, and they exchange messages. Each exchange of messages reveals something about the situation, or, in technical terms, it reduces the size of the relevant Kripke structure or Aumann structure. An agent who has seen some events but not others can make guesses as to what other events *could* have taken place and it knows some fact ϕ iff ϕ would be true *regardless* of how the unseen events went. This framework is used in both [9, 7]. .

4 An abstract model

We now present an abstract extensional presentation of a communication system in which the system is described as a set of *global histories*, each of which represents one possible system evolution given by a sequence of global events. For each system, the set of *agents* that participate in its computations is assumed to be a fixed finite set. Similarly, for each system, the set of possible global events is fixed.

For convenience, we fix $n > 0$, and consider only systems with agents from $[n] = \{1, 2, \ldots, n\}$, and events form a fixed (possibly infinite) set E. E^* is the set of all finite sequences over E and E^ω is the set of all infinite sequences over E; we will let $h, h', \ldots$ range over the former, and $H, H', \ldots$ over the set $E^* \cup E^\omega$. Let $H \preceq H'$ denote that H is a finite prefix of H'. We write $h; H'$ or just hH' to denote the concatenation of the finite history h with the possibly infinite history H'. When H is infinite or of length $\geq k$, we let H_k denote the finite prefix of H consisting of the first k elements. For a set $\mathcal{H}$, let $\mathcal{P}(\mathcal{H})$ denote the set $\{h \mid h \preceq H \text{ for some } H \in \mathcal{H}\}$ containing all finite prefixes of sequences in $\mathcal{H}$.

.

The set of events E typically consists of actions by agents in the system (including the sending and receipt of messages), but may also include other events (perhaps due to actions of the environment) that affect the knowledge of agents. We do not have a specific syntax of messages here, but choose to identify the message with the event that denotes its sending or receipt; in this sense, when we talk of the meaning of a message, we are referring to what the sending (receiving) of that message (at a specific time, in a context) signifies to the sender (receiver). Thus we are really discussing the semantics of event occurrences as perceived by agents in the system.

Definition 4.1 *A* **system** *is a tuple $S = (\mathcal{H}, E_1, \ldots, E_n)$, where $\mathcal{H} \subseteq E^\omega$ (our* protocol *) is the set of all (infinite) possible global histories of S, and for $i \in [n]$, $E_i \subseteq E$ is the set of* local events *of agent i (not necessarily disjoint from E_j for $j \neq i$).*

.

The role of the protocol is to limit the possible global histories which any agent may consider. It is this limitation on what can happen globally that permits an agent to make inferences from locally observed events to non-observed events.

Local histories are got by 'projecting' global histories to local components. For $i \in [n]$, let $\lceil_i : \mathcal{P}(E^\omega) \to E^*$ be the *projection map* for i, such that $\lceil_i(H)$ is obtained by mapping each event in E_i into itself, and each event from $E - E_i$ into the null string. $\mathcal{H}_i \stackrel{\text{def}}{=} \{\lceil_i(h) \mid h \in \mathcal{P}(\mathcal{H})\}$ is the set of local histories of i.

The local history of process i corresponding to global history H at time k consists simply of those events from H_k which are *seen* by process i. Thus if $h_1 \preceq h_2 \preceq H \in \mathcal{H}$, then $\lceil_i(h_1) \preceq \lceil_i(h_2)$ as well. In particular, if h is the local history of process i at some stage, and event e visible to i takes place next (that is, $e \in E_i$), then $h; e$ will be the resulting history. If e is not visible, then the new history would still be h.

.

Definition 4.2 *Let $h, h' \in \mathcal{P}(\mathcal{H})$. For $i \in [n]$, define $h \sim_i h'$ iff $\lceil_i(h) = \lceil_i(h')$.*

$\sim_i$ is an equivalence relation, and it gives the *indistinguishability relation for* i. We can consider this relation as giving the information partition for i in the system S; that is, given the information available to i, the histories h and h' cannot be distinguished. i can only know properties *common* to h, h'.

.

The properties of such systems can be studied in a logical language. Let L be a language which has formulae expressing (time dependent) properties of global histories. Then we can write $H, k \models A$, for A belonging to L, to mean that the history H satisfies formula A at time k. If the truth value of A does not depend on k, then it is timeless. If A has the property that once true it remains true, then it is persistent. We expand L to a larger language LK by closing under boolean connectives and operators K_i. Thus if A is a formula of LK and i is a process, then $K_i(A)$, meaning i knows A, is also in LK. We can then define $H, k \models K_i(A)$ to hold if for all m and all $H' \in \mathcal{H}$, if

$H'_m \sim_i H_k$ then $H', m \models A$. What the process i knows at time k depends on its local history. Moreover, the laws of logic $LK5$ (the $S5$ version of the logic of knowledge) are valid.

.

For definiteness, we fix a specific language L so that the semantics of $H, k \models A$ is also fixed. Since the basic elements of the model are sequences, a linear time temporal logic suggests itself. Let $P = \{p_0, p_1, \ldots\}$ be a countable set of atomic propositions. Formally, the syntax of the logic is given by:

$$\alpha, \beta \in \mathcal{L}_0 ::= p \in P \mid \neg\alpha \mid \alpha \vee \beta \mid \bigcirc\alpha \mid \alpha \mathrm{U} \beta \mid K_i\alpha$$

.

A **model** is a pair $M = (S, V)$, where $V : \mathcal{P}(\mathcal{H}) \to 2^P$ is a valuation map on finite prefixes of global histories which gives the truth values of some atomic predicates at the states. We can now inductively define the notion $H, k \models \alpha$, for $H \in \mathcal{H}$, $k \geq 0$ and $\alpha \in \mathcal{L}_0$:

1. $H, k \models p$ iff $p \in V(H_k)$, for $p \in P$.
2. $H, k \models \neg\alpha$ iff $H, k \not\models \alpha$.
3. $H, k \models \alpha \vee \beta$ iff $H, k \models \alpha$ or $H, k \models \beta$.
4. $H, k \models \bigcirc\alpha$ iff $H, k+1 \models \alpha$.
5. $H, k \models \alpha \mathrm{U} \beta$ iff there exists $m \geq k$ such that $H, m \models \beta$ and for all $\ell : k \leq \ell < m$, $H, \ell \models \alpha$.
6. $H, k \models K_i\alpha$ iff for all $m \geq 0$, for all $H' \in \mathcal{H}$ such that $H_k \sim_i H'_m$, $H', m \models \alpha$.

.

If each agent i has its own notion $\mathcal{H}_i$ of what the protocol is, then we would use the definition: $H, k \models K_i\alpha$ iff for all $m \geq 0$, for all $H' \in \mathcal{H}_i$ such that $H_k \sim_i H'_m$, $H', m \models \alpha$. In that case iterated knowledge modalities involving different agents will become problematic. For instance agent i may not know what set $\mathcal{H}_j$ is, see for instance [12]. But certain situations where the protocol itself is not common knowledge can only be modelled in this way.

Other temporal connectives can be defined from U. For instance $\Diamond\alpha$, meaning 'α will hold sometime in the future,' can be defined as $\mathit{true}\mathrm{U}\alpha$. Also, since the truth value of a formula of the form $K_i\alpha$ at H, k depends only on $h = \lceil_i(H_k)$, we shall occasionally abuse language and write $h \models K_i(\alpha)$ when we mean $H, k \models K_i(\alpha)$. See [HV] for a general study of the relationship between knowledge and time.

.

The formula α is said to be *satisfiable* if there exists a model M, a global history $H \in \mathcal{H}$ in M and $k \geq 0$ such that $M, k \models \alpha$. α is said to be *valid* iff $\neg\alpha$ is not satisfiable. The following laws of the logic $LK5$ are easily seen to be valid:

- $K_i(\alpha \supset \beta) \supset (K_i\alpha \supset K_i\beta)$.
- $K_i\alpha \supset \alpha$.
- $K_i\alpha \supset K_iK_i\alpha$.
- $\neg K_i\alpha \supset K_i\neg K_i\alpha$.

There are of course other laws which connect K_i with the temporal connectives. However, we shall not attempt to give a complete axiomatization here.

.

We are now ready to present the knowledge-based semantics of messages in a system S. For our purpose, a message is simply an element $m \in E_i$. Of

course, we normally think of a message as a *receive* event, caused by some previous *send* event. However, there is no harm in allowing knowledge to be obtained from all events – even the ticking of a clock conveys the information that time has passed.

Definition 4.3 *Let* $i \in [n]$ *and* $m \in E_i$.
$Sem_S(i,m) \stackrel{\text{def}}{=} \{(\alpha,\beta) \mid \forall h; m \in \lceil_i(\mathcal{H}) : (h \models \alpha \rightarrow h; m \models K_i\beta)\}$

.

Intuitively, (α,β) is part of the meaning of m if whenever event m takes place, then if α was *true* before, i always *knows* β after m has taken place. Thus the meaning of m is given in terms of properties expressible in the logical language, which is a meta-language. This approach, of giving semantics for messages (themselves semantic entities here, in the sense that they are elements of models and are not syntactic objects in a specified language) using syntactic means (formulas in a logical language) may seem curious. But this has definite and interesting implications, and allows for us to consider communication between parties who do not share a language (as we will see in a later section), or even signals that do not come from any apparently recognizable language. The use of a logical language here emphasizes that the discussion of what a communication means to those involved in the communication is limited only by the level of reasoning carried out at the meta-level.

.

$Sem_S(i,m)$ is a subset of $\mathcal{L}_0 \times \mathcal{L}_0$, and every pair (α,β) in it can be seen as a *view transformer* for agent i: if knowledge of α is part of the view that i has of the global system state before the communication event, then α was also true before the communication and hence the knowledge of β is part of its view after the communication.

It is easy to see that the definition can be generalized to $Sem_S(i,h)$, where

$h \in E_i^*$: $Sem_S(i,h) \stackrel{\text{def}}{=} \{(\alpha,\beta) \mid \text{for all } h'; h \in \lceil_i\mathcal{H}\text{: } (h' \models \alpha \rightarrow h'; h \models K_i\beta)\}$. We will drop the subscript S when the context is clear. Moreover, we write $\beta \in Sm(i,h)$ to mean that $(true,\beta) \in Sem(i,h)$, i.e. that h allows i to learn β without any presupposition. This allows us also to speak of the meaning of h as a set of formulas rather than as a set of pairs of formulas. The notation $Sm(i,m)$ for $m \in E_i$ is similarly understood. Note that in general $Sem(i,m)$ will be richer than $Sm(i,m)$ and not reducible to it.

. Note that an event which is invisible to process i results in no learning. For

if h, $h;e$ are finite global histories and e is invisible to i then $\lceil_i(h) = \lceil_i(h;e)$ and thus $h \sim_i h;e$. Thus the same K_i formulae will be true at both histories.

It is easy to see that $Sem_S(i,h)$ is closed under conjunction and weakening, i.e., if $(\alpha,\beta_1) \in Sem_S(i,h)$ and $(\alpha,\beta_2) \in Sem_S(i,h)$ then $(\alpha,\beta_1 \wedge \beta_2) \in Sem_S(i,h)$ and for all γ, $(\alpha,\beta_1 \vee \gamma)$ is also in $Sem_S(i,h)$. Moreover, since knowledge of β implies the truth of β, if $(\alpha,\beta) \in Sem_S(i,h)$ and $(\beta,\gamma) \in Sem_S(i,h')$ then
$(\alpha,\gamma) \in Sem_S(i,h;h')$.

.

Proposition 4.4 *Suppose we are given two systems $S = (\mathcal{H}, E_1, \ldots, E_n)$ and $S' = (\mathcal{H}', E_1, \ldots, E_n)$ which are identical in their structure except that $\mathcal{H}' \subseteq \mathcal{H}$. Let α, β be two formulas such that α is knowledge free (not containing any operators K_i) and all occurrences of the operators K_i in β are positive, i.e. occur under no negation signs or under an even number of them. Then if (α,β) is in $Sem_S(i,m)$, it is also in $Sem_{S'}(i,m)$.*

The proof is straightforward using the fact that the quantification over histories of S' which is required for interpreting the modalities K_i is always over a smaller set and hence more likely to be true. Thus given a set of conventions which restrict the set of possible histories, more information can be acquired through a message. This of course need not hold with negative knowledge.

It may happen that in a larger $\mathcal{H}$, after message m is received, some $\neg K_i(\beta)$ holds, but that with a smaller $\mathcal{H}'$, $K_i(\beta)$ does hold and hence $\neg K_i(\beta)$ fails.

.

For a very simple example, if you tell me β but the global histories in $\mathcal{H}$ allow the possibility of your lying, then I do not know β, even if you are in fact sincere. But if we are restricted to global histories in which you never lie, then I *will* know β.

.

[7] discusses agents who are connected along some graph, and knowledge can move only along the edges of a graph. Thus if agent i is not connected to agent j, then i cannot directly obtain information from j, but might get such information via a third agent k, as in fact Novak got some information from Judith Miller. Such edges may be approved or disapproved, and if information transfer took place along a disapproved edge, then that could be cause for legal sanctions, not because harm had occurred, but because harm *could* occur and the algorithm was no longer secure.

It is shown in [7] that the graph completely determines the logical properties of possible states of knowledge, and vice versa. Indeed, an early version of that paper already discussed the Plame case before it hit the headlines..

In [9] we consider how obligations arise from knowledge. We consider the following examples:

Example 1: Uma is a physician whose neighbour is ill. Uma does not know and has not been informed. Uma has no obligation (as yet) to treat the neighbour.

Example 2: Uma is a physician whose neighbour Sam is ill. The neighbour's daughter Ann comes to Uma's house and tells her. Now Uma does have

an obligation to treat Sam, or perhaps call in an ambulance or a specialist.

Example 3: 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..

In such cases, when an agent cannot herself take a requisite action, it is incumbent upon her to provide such information to the agent who *can* take such action. Or, as in the case of the hospital, the agent has an obligation not only to act, but also to gather knowledge so as to be *able to act* when the occasion arises. A milder example of such situations consists of requiring homeowners to install fire alarms.

Again the semantics from [16] is used. Various possible sequences of events are possible, depending on the actions taken by the agents. Some of these sequences are better than others, and some, indeed are disastrous, as when a patient is not treated for lack of information. It is shown how *having information* creates obligations on the agents, and also how the need to *convey information* arises, when one knows that an agent who could carry out some required action lacks the requisite information.

.

Consider the following events $S = Sam\ is\ sick.\ T = Uma\ treats\ Sam.\ M = Ann\ gives\ a\ message\ to\ Uma\ that\ Sam\ is\ ill.$

Now we consider histories H_1, H_2, H_3, H_4, H_5. In H_1, nothing happens and no one is sick. In H_2, Sam is not sick but Uma attempts to treat a healthy Sam. In H_3, Sam is sick but Ann does not inform Uma. In H_4, Sam is Sick and Ann informs Uma who does not treat him. Finally, in H_5, Sam is sick, Ann informs Uma and Uma treats Sam.

$H_4 < H_3 < H_5 < H_2 < H_1$, so H_1 is best, and H_4 is worst.

We have, $H_1 \sim_u H_3$. So unless Uma hears from Ann, she does not know that

Sam is sick and should not try to treat him. On the other hand, once she does hear from Ann, she has to choose between H_4 and H_5 and she should choose H_5, i.e., treat Sam. We can also show that knowing Uma's choices, Ann has the obligation to inform Sam.

.

5 Summary

We have given examples of situations where knowledge transfer and algorithmic structure can affect or even determine the sorts of social algorithms which are possible. In the full paper we will give some technical results.

.

References

[1] R. Aumann, "Agreeing to disagree", *Annals of Statistics*, **4** (1976) 1236-1239.

[2] Steven Brams and Alan Taylor, *The Win-Win Solution: guaranteeing fair shares to everybody*, Norton 1999.

[3] Fagin, R., J. Halpern, Y. Moses and M. Vardi, *Reasoning about Knowledge*, MIT Press 1995.

[4] J. Geanakoplos and H. Polemarchakis, "We can't disagree forever," *J. Economic Theory.*

[HV] Halpern, J., and M. Vardi, The complexity of reasoning about knowledge and time, in *J. Computer and System Sciences*, **38** 195-237, 1989.

[5] Jaakko Hintikka, *Knowledge and Belief: an introduction to the logic of the two notions*, Cornell University press, 1962.

[6] D. Lewis, *Convention, a Philosophical Study*, Harvard U. Press, 1969.

[7] Eric Pacuit and Rohit Parikh, "Reasoning about Communication Graphs," Presented at Augustus de Morgan Workshop: Interactive Logic: Games and Social Software, 2006 (ADMW 2006). To appear in the *proceedings of ADMW.*

[8] Eric Pacuit and Rohit Parikh "Social Interaction, Knowledge, and Social Software," In *Interactive Computation: The New Paradigm*, Springer-Verlag 2006. Editors Dina Goldin, Scott Smolka and Peter Wegner.

[9] Eric Pacuit, Rohit Parikh and Eva Cogan, "The Logic of Knowledge Based Obligation," *Knowledge, Rationality and Action*, a subjournal of *Synthese*, 149(2), 311 – 341, 2006.

[10] R. Parikh "The Logic of Games and its Applications," *Annals of Discrete Math.*, 24 (1985) 111-140.

[11] R. Parikh, "Social Software," *Synthese*, **132**, Sep 2002, 187-211.

[12] R. Parikh, "Levels of knowledge, games, and group action," in *Research in Economics*, **57**, (2003) 267-281.

[13] R. Parikh and P. Krasucki, "Communication, Consensus and Knowledge," *J. Economic Theory* **52** (1990) pp. 178-189.

[14] R. Parikh, Laxmi Parida and Vaughan Pratt "Sock Sorting," appeared in a volume dedicated to Johan van Benthem, University of Amsterdam, August 99, reprinted in *Logic J. of IGPL*, vol 9 (2001).

[15] R. Parikh and R. Ramanujam "Distributed Processing and the Logic of Knowledge," in *Logics of Programs* Proceedings of a Conference at Brooklyn College, June 1985, Springer Lecture Notes in Computer Science #193, pp. 256-268.

[16] Parikh, R. and Ramanujam, R., A knowledge based semantics of messages, in *J. Logic, Language, and Information*, **12**, pp. 453 - 467, 2003.

[17] Donald Shoup, "Gone Parkin'," Op-Ed page, *The New York Times*, March 29, 2007.