Topics in Social Software

Information in Strategic Situations

(Chapter 1)

Eric Pacuit

April 15, 2005

Chapter 1

Introduction

Donald Knuth begins his remarkable series of books, *The Art of Computer Programming*, with the statement "The notion of an algorithm is basic to all computer programming, and so we should begin with a careful analysis of this concept." In other words, in order to understand the complex behavior of a computer, one needs a systematic and rigorous analysis of algorithms, or procedures. The key idea behind *social software* is to apply this simple idea to social situations. That is, a systematic and rigorous analysis of *social procedures* can help us understand social interactions and may lead to a more "efficient" society. This idea was put forward by Rohit Parikh [Par95], and has recently gained the attention of a wide range of research communities, including computer scientists, game theorists and philosophers. 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.

1.1 What is Social Software?

Starting with [Par95] and more recently in [Par02a, Par02b, Par01], Parikh defines social software by way of various illustrative examples. Essentially, there are two ways in which a procedure can fit into the social software paradigm. First of all, a procedure may be truly social in that several agents are required even in the execution of the procedure. Standard examples are voting procedures, such as plurality voting or approval voting, or fair division algorithms, such as adjusted winner or the many cake-cutting algorithms. Secondly, even if a procedure does not *require* a group for its execution, it may still fit into the social software paradigm. These are procedures set up by society and intended to be performed by single agents within the context of a group of agents. Examples include proce-

dures that universities set up that students must follow in order to drop a class or the procedures hospitals set up to ensure the necessary flow of information from a patient to a doctor.

From the point of view of someone designing a social procedure, as soon as beliefs and utilities can be attributed to the agent(s) executing the procedure, the procedure should be thought of as social software. After all, when designing computer software, programmers do not worry that the computer may suddenly not "feel like" performing the next step of the algorithm. But in a setting where agents have individual preferences, such considerations must be taken into account. In fact, this suggests a third way in which procedures can be analyzed within the social software paradigm - individual agents executing procedures in isolation. For example, an agent following a recipe in order to make peanut-butter chocolate chip cookies. However, from this point of view, certain philosophical questions about the nature of procedures, or algorithms, and human knowledge and beliefs become much more important. In this thesis, the fact that a group of agents is somehow involved in the execution of a social procedure will play an essential role. As a consequence, the study of social software draws on results not only from artificial intelligence, distributed computing and philosophical logic but also from game theory, especially mechanism design, and economics.

So far we have only explained the type of situations we have in mind and have not yet provided an adequate *definition* of social software. We will now attempt to rectify this situation. 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: modeling social situations, developing a theory of correctness of social procedures and designing social procedures. The next three subsections will describe these areas in more detail.

1.1.1 Models of Social Situations

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. If one wants a careful and rigorous analysis of social procedures, one needs to begin with a realistic model of multi-agent interaction. The search for such models has occupied researchers in a number of different disciplines including (but not limited to), game theory, philosophy, artificial intelligence and distributed computing. What is needed from the social software point of view are formal models in which our intuitions about social procedures can be refined and tested.

It is important to be clear about exactly what is being proposed. Perhaps

it is too much to ask for a general theory which explains all social interactions, i.e., a "theory of everything" for the social sciences. If at all possible, such a theory would require collaboration among a vast array of research communities including psychologists, biologists, cognitive scientists and so on. What *is* being developed is a collection of logical systems intended to be used to formalize multi-agent interactive situations relevant for the analysis of social procedures. These frameworks are developed from different points of view and are governed by different assumptions about the agents involved.

There are many ways one could formalize social situations in which the agents are assumed to be executing some procedure. In this thesis, the role of information and its dynamics in strategic multi-agent situations is a central issue. To be more precise, we focus on the following two issues:

1. **Knowledge, Actions and Obligations:** It is natural to assume that the agents' choices, at least partially, depend on their current states of information. This is particularly important when reasoning about what agents ought to do. Certainly an agent cannot be faulted for not performing an action whose need it did not know about. How should this dependency of actions on knowledge be modeled?

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. How should information update be represented? How does this affect the models of knowledge and belief?

1.1.2 A Theory of Correctness of Social Procedures

Just as one can prove that a certain implementation of a sorting algorithm is correct, perhaps one can prove that a certain piece of social software is correct. In computer science, it is often convenient to view a computational procedure, or a program, as a relation on a set of states, where a state can be thought of as a function that assigns a value to every possible variable and a truth value to all propositions. This approach was first proposed by Pratt ([Pra76]) and is based on the work of Floyd ([Flo67]) and Hoare ([Hoa69]). Harel, Kozen and Tiuryn ([HKT00]) provide a very thorough discussion of computational procedures from this point of view. The first step towards a formal logic of social procedures influenced by this analysis of computational procedures is Parikh's game logic ([Par83, Par85]). Game logic is intended to formalize situations where agents have directly opposing preferences. Marc Pauly took the analysis one step further in his dissertation, *The Logic of Social Software* [Pau01], in which he developed a logic

for reasoning about coalitions [Pau02a, Pau02b]. Recently, Pauly has developed a formal framework in the style of Hoare [Hoa69] for proving correctness of social procedures [Pau].

However, none of the frameworks discussed above provide an explicit representation of agents' knowledge. Finding an appropriate representation is an important step towards a formal theory of correctness of social procedures. As discussed above, one of the main aspects of social procedures is that a group of agents is somehow involved. In this multi-agent setting it is natural to assume that the agents do not all have exactly the same information. And so issues about how each agent represents the other agents' information becomes important. In particular, notions such as common knowledge, distributed knowledge, and other levels of knowledge[1] are very relevant when trying to formalize correctness conditions of many social procedures.

To illustrate the above point, consider the fair division algorithm *adjusted winner* [BT96]. Adjusted winner is an algorithm to fairly distribute divisible goods among two people. The procedure is discussed in detail in [BT96]. Instead of providing a formal description of the procedure, we will look at an illustrative example. More theoretical and practical information can be found in the books [BT96, BT99]. Suppose Ann and Bob are dividing three goods: A, B, and C. Adjusted winner begins by giving both Ann and Bob 100 points to divide among the three goods. Suppose that Ann and Bob assign points according to the following table.

Item	Ann	Bob
A	$\underline{10}$	5
B	$\underline{65}$	45
C	25	$\underline{50}$
Total	100	100

The second step of the procedure is to give A and B to Ann since she assigned more points to those items and item C to Bob. However this is not an equitable outcome since Ann has received 75 points while Bob only received 50 points. We must transfer some of Ann's goods to Bob. However, even giving all of item A to Bob will not create an equitable division since Ann now has 65 points, but Bob has only 60 points. In order to create equitability, we must transfer part of item B from Ann to Bob. Let p be the proportion of item B that Ann will keep. Then

$$65p = 100 - 45p$$

[1] See [PK92, Par03] for more on the application of levels of knowledge *below* common knowledge to game theoretic situations.

yielding $p = 100/110 = 0.9090$, so Ann will keep 91% of item B and Bob will get 9% of item B. Thus both Ann and Bob receive 59.09 points. This allocation (Ann receives 91% of item B and Bob receives all of item A and item C plus 9% of item B) is *envy-free*, *equitable* and *efficient*. See [BT96] for a formal definition of these properties and proofs that the adjusted winner procedure satisfies these properties. What we are interested in is whether Ann can improve her total allocation by misrepresenting her preferences. It turns out that she can improve her allocation. If Ann announces that her allocation is 6 points for A, 55 points for B and 39 points for C, then the adjusted winner algorithm will give all of B to Ann and all of A and C to Bob. This results in Ann receiving a total of 65 points (according to her original preferences) while Bob receives 55 points. Now this type of deception is only possible in the extreme case that Ann knows Bob's preferences, but *Bob does not know Ann's preferences*. Indeed, Brams and Taylor successfully argue in [BT96] that 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. The point is that part of showing that an *implementation* of adjusted winner is correct must take into account that the each agent's preference must be kept private, or at least that the agents cannot take advantage of whatever information they have about the other agent's preferences. In other words, given the results from Brams and Taylor that adjusted winner is efficient, envy free and equitable, showing that an implementation of adjusted winner is correct reduces to showing that a particular level of knowledge among the agents is maintained (i.e., each agent's preference is kept private).

1.1.3 Designing Social Procedures

An important test of any theory is how well it can be applied to real life situations. There are a wealth of social procedures described in the game theory literature. For instance, see [BT96] for a discussion of a number of different fair division algorithms. Attempts to formalize these procedures create a wide range of interesting problems for logicians. For example, see [PS04, Sal05] for a logic that axiomatizes the concept of majority. It will be very interesting to see if logical analyzes can suggest refinements of existing social procedures or help create new social procedures.

1.2 Social Software for the Game Theorist

Starting with von Neumann and Morgenstern, and later with the seminal work of Nash, game theorists have developed elegant mathematical theories that can formalize situations similar to the one described above. Mechanism design (or implementation theory) studies situations in which the agents' preferences are given together with a desired set of outcomes, and asks what interactive situation (a game) can be designed in which the outcomes can be achieved assuming the agents act according to their given preferences. See [OR94] Chapter 10 for more information. Essentially, a piece of social software is created. Pauly and Wooldridge [PW] and Halpern [Hal03] independently have recently argued that formal logical systems can be useful in this setting.

Evidence of the usefulness of a rigorous analysis of social procedures can already be seen in many areas of economics and social choice theory. Classic results such as Arrow's Theorem or the Gibbard-Satterthwaite Theorem [Gib73, Sat73, Sat75] have had profound effects on social choice theory and voting theory. A more concrete example is the game-theoretic analysis of the procedure used by King Solomon in the well-known biblical story. In this story, King Solomon is faced with two women each claiming that a baby is her own child. Solomon threatens to cut the baby in half causing one of the mothers to rescind her claim of motherhood; thus revealing herself as the true mother. However a formal analysis of this procedure, first pointed out by Glazer and Ma in 1989 [GM89], reveals a mistake in Solomon's procedure. It is possible for the false mother to outwit Solomon by misrepresenting her actual preference and claim, as the true mother would, that the baby should be given to the other woman. Glazer and Ma suggest a small refinement of the procedure (involving the use of money) creating a "strategy-proof" procedure. What is missing from these analyzes is an explicit and rigorous description of what it means for a social procedure to be correct. An important objective of social software is to fill this gap.

At this point, we have only argued for a formal and rigorous analysis of social procedures and have said nothing about why *logic* should be used for such an analysis. Presumably, many researchers will agree that rigor is important when analyzing social procedures, but may hesitate to say that a logical analysis is also important. Using logic to analyze social procedures is an important part of social software research. Certainly, it is hard to argue that logic is not an important tool if the goal is to mechanize the analysis of social procedures. But what if the goal is not a computer program, but rather the analysis of a social procedure itself? Can a logical analysis still be of use? Answering this question raises a number of very interesting an important issues which have been addressed by a number of different authors (for example see [Par87, Sta91, Sta99, Bac97, Bon04,

vB01, Aum99, Rub00]). A complete answer to this question will be an interesting digression, but a digression nonetheless. Instead we let Bacharach have the final say:

> Game theory is full of deep puzzles, and there is often disagreement about proposed solutions to them. The puzzlement and disagreement are neither empirical nor mathematical but, rather, concern the meanings of fundamental concepts ('solution', 'rational', 'complete information') and the soundness of certain arguments...Logic appears to be an appropriate tool for game theory both because these conceptual obscurities involve notions such as reasoning, knowledge and counterfactuality which are part of the stock-in-trade of logic, and because it is a prime function of logic to establish the validity or invalidity of disputed arguments. — M.O.L. Bacharach [Bac97]

1.3 Social Software for the Computer Scientist

The existence of more and more autonomous programs has prompted the study of *computational* mechanism design which applies economic principles to the design of multi-agent systems. A formal theory of correctness of social procedures can be used to develop computational tools that can verify interactive multi-agent protocols, such as online auctions. See [DPJ03, San03, STar] for discussions of the relevant issues.

For example, if I want to fly to India, instead of searching for the best deal and purchasing the ticket myself, I may give a set of constraints to an Internet agent (see [Woo00] for issues related to designing such autonomous computational agents), and then send the agent out to find me the best possible price (given my constraints) and return with the purchased ticket. Of course, at the other end the airlines may also have an autonomous agent designed to sell tickets at a certain price. Now, these agents will interact and bargain according to some predefined procedure, or protocol. How do we know that the procedure the agents follow, which after all is just another piece of code, actually implements the intended social interaction? In other words, the interaction of the internet agents is intended to somehow mimic a real-life interaction between a buyer and a seller. A formal theory of correctness of social software could be used to determine whether the procedure under consideration, in this case the procedure that the seller agent and the buyer agent use to come to a deal, is "sound" and "complete" for its intended interpretation. By "sound", I mean that by following the rules of the procedure no unintended consequences are generated. For example, the

buyer should not be able to leave with a ticket without spending any money. "Completeness" is slightly more complicated. Suppose that P is some procedure intended to mimic some social interaction, say S. Now there are many possible consequences of the social interaction S. These consequences may be described by propositional formulas, such as "the buyer has a ticket to India", but also may be knowledge-theoretic in nature, such as the buyer believes that it got the best deal. The procedure P would be "complete" for situation S if all such consequences could be achieved by the agents.

Finally, we point to another application of social software relevant for both computer scientists and game theorists. The study of rational agents is central to both game theory and artificial intelligence. Simply put, an agent is an entity situated in an environment who is capable of performing actions that somehow change the environment; a *rational agent* chooses actions that are in its own best interests. In [vdHW03], van der Hoek and Wooldridge discuss the importance and difficulty of developing a logic of rational agency. The logical systems developed in this thesis deal with many of the same questions as those posed in [vdHW03].

1.4 Overview

This thesis uses techniques from modal logic, especially epistemic logic, and game theory to develop formal models appropriate for the analysis of social software. The basics of modal logic and game theory will be assumed throughout the thesis, see [BdRV02] for more information on modal logic and [OR94] for more information on game theory.

The basic formal framework used is a history based model. The idea is that each agent has a set of possible actions, or choices, and at each moment some event takes place. In computer science, history based models have been used to model computations in a distributed environment. See [FHMV95] for a thorough discussion. In the game theory literature, the history based models are called extensive games. The reader is referred to the textbook [OR94] for a discussion of extensive games. Chapter 2 provides an introduction to this framework.

Chapter 3 reports on joint work with Rohit Parikh and Eva Cogan. Starting with the intuition that agents cannot be expected to perform actions they are unaware of, a multi-agent logic of knowledge, action and obligation is developed. Various deontic dilemmas are studied that illustrate the dependency of an agent's obligation on its knowledge.

Chapter 4 is the result of joint work with Rohit Parikh. In [PP05], agents are assumed to have some private information at the outset, but may refine their information by acquiring information possessed by other agents, possibly via yet

other agents. A multi-agent modal logic with knowledge modalities and a modality representing communication among agents is introduced and shown to be decidable.

Chapter 5 looks at voting theory from the social software point of view. This chapter is based on joint work with Samir Chopra and Rohit Parikh [CPP04].

Finally, in Chapter 6 we conclude and point to further directions of research.

Bibliography

[Aum99] R. Aumann. Interactive epistemology I: Knowledge. *International Journal of Game Theory*, 28:263–300, 1999.

[Bac97] M. O. L. Bacharach. The epistemic structure of a theory of a game. In M.O.L. Bacharach, L.A¿ Gerard-Varet, P. Mongin, and H.S. Shin, editors, *Epistemic Logic and the Theory of Games and Decisions*, pages 303 – 344. Kluwer Academic Publishers, 1997.

[BdRV02] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Campbridge University Press, 2002.

[Bon04] Giacomo Bonanno. A characterization of von neumann games in terms of memory. *Synthese*, 139(2):237–256, March 2004.

[BT96] S. J. Brams and A. D. Taylor. *Fair Division: From cake-cutting to dispute resolution*. Cambridge University Press, 1996.

[BT99] S. J. Brams and A. D. Taylor. *The Win-Win Solution*. W. W. Norton and Company, 1999.

[CPP04] Samir Chopra, Eric Pacuit, and Rohit Parikh. Knowledge-theoretic properties of strategic voting. In Jose Julio Alferes and Joao Leite, editors, *Proceedings of JELIA 2004*, Lecture Notes in Artificial Intelligence, pages 18–30. Springer, 2004.

[DPJ03] R. K. Dash, D. Parkes, and N. R. Jennings. Computational mechanism design : A call to arms. *IEEE Intelligent Systems*, 18(6):40–47, 2003.

[FHMV95] R. Fagin, J. Halpern, Y. Moses, and M. Vardi. *Reasoning about Knowledge*. The MIT Press, 1995.

[Flo67] R. W. Floyd. Assigning meanings to programs. *Proc. Symp. Appl. Math*, 19:19–31, 1967.

[Gib73] Allan Gibbard. Manipulation of Voting Schemes: A General Result. *Econometrica*, 41(4):587–601, 1973.

[GM89] J. Glazer and C. A. Ma. Efficient allocation of a 'prize' — king solomon's dilemma. *Games and Economic Behavior*, 1:222 – 233, 1989.

[Hal03] Joseph Halpern. A computer scientist looks at game theory. *Games and Economic Behavior*, 45(1):114 – 131, 2003.

[HKT00] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.

[Hoa69] C. A. R. Hoare. An axiomatic basis for computer programming. *Comm. Assoc. Comput. Mach.*, 12, 1969.

[OR94] M. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press, 1994.

[Par83] Rohit Parikh. Propositioanl logics of programs: new directions. In M. Karpinski, editor, *Foundations of Computation Theory*, number 158 in LNCS, pages 347 – 359. Springer, 1983.

[Par85] Rohit Parikh. The logic of games and its applications. In M. Karpinski and J. van Leeuwen, editors, *Topics in the Theory of Computation*, volume 24 of *Annals of Discrete Mathematics*. Elsevier, 1985.

[Par87] R. Parikh. Knowledge and the problem of logical omniscience. volume ISMIS-87, pages 432 – 439. North Holland, 1987.

[Par95] Rohit Parikh. Language as social software (abstract). In *International Congress on Logic, Methodology and Philosophy of Science*, page 415, 1995.

[Par01] Rohit Parikh. Lanuage as social software. In S. Shieh J. Floyd, editor, *Future Pasts: The Analytic Tradition in Twentieth Century Philosophy*, pages 339–350, 2001.

[Par02a] Rohit Parikh. Social software. *Synthese*, 132:187–211, September 2002.

[Par02b] Rohit Parikh. Towards a theory of social software. In *Proceedings of DEON 2002*, number 132, pages 187–211, September 2002.

[Par03] Rohit Parikh. Levels of knowledge, games, and group action. *Research in Economics*, 57(3):267 – 281, 2003.

[Pau] Marc Pauly. Programming and verifying subgame perfect mechanisms. to appear in the *Journal of Logic and Computation*.

[Pau01] Marc Pauly. *Logic for Social Software*. Illc dissertation series 2001-10, University of Amsterdam, 2001.

[Pau02a] Marc Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12(1):149 – 166, 2002.

[Pau02b] Marc Pauly. On the complexity of coalitional reasoning. *International Game Theory Review*, 4(3):237 – 254, 2002.

[PK92] Rohit Parikh and Paul Krasucki. Levels of knowledge in distributed computing. *Sadhana - Proc. Ind. Acad. Sci.*, 17:167 – 191, 1992.

[PP05] Eric Pacuit and Rohit Parikh. The logic of communication graphs. In J. Leita, A. Omicini, P. Torroni, and P. Yolum, editors, *Proceedings of DALT 2004*, Lecture Notes in AI, pages 256 – 269. Springer, 2005.

[Pra76] V. R. Pratt. Semantical considerations on floyd-hoare logic. In *Proc. 17th Symp. Found. Comput. Sci.*, IEEE, pages 109 – 121, 1976.

[PS04] Eric Pacuit and Samer Salame. Majority logic. In Didier Dubois, Christopher A. Welty, and Mary-Anne Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 598–605. AAAI Press, June 2 - 5 2004.

[PW] Marc Pauly and Mike Wooldridge. Logic for mechanism design – a manifesto. Unpublished manuscript.

[Rub00] Ariel Rubinstein. *Economics and Language*. Cambridge University Press, 2000.

[Sal05] Samer Salame. *Majority Logic*. PhD thesis, CUNY Graduate Center, 2005.

[San03] Tuomas Sandholm. Making markets and democracy work: A story of incentives and computing. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-03)*, pages 1649–1671, 2003.

[Sat73] Mark Satterthwaite. *The Existence of a Strategy Proof Voting Procedure: a Topic in Social Choice Theory*. PhD thesis, University of Wisconson, 1973.

[Sat75] Mark Satterthwaite. Strategy-proofness and Arrow's Conditions: Existence and Correspondence Theorems for Voting Procedures and Social Welfare Functions. *Journal of Economic Theory*, 10(2):187–217, 1975.

[Sta91] R. Stalnaker. The problem of logical omniscience, I. *Synthese*, 89:425 – 440, 1991.

[Sta99] R. Stalnaker. *Context and Content*, chapter The Problem of Logical Omniscience, II. Oxford: Clarendon Press, 1999.

[STar] Y. Shoham and M. Tennenholtz. Non-cooperative computing: Boolean functions with correctness and exclusivity. *Journal of Theoretical Computer Science*, to appear.

[vB01] J. van Benthem. Games in dynamic epistemic logic. *Bulletin of Economic Research*, 53:216 – 248, 2001.

[vdHW03] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *L. J. of the IGPL*, 11(2):135 – 159, 2003.

[Woo00] Michael Wooldridge. *Reasoning about Rational Agents*. The MIT Press, 2000.