ELSEVIER Artificial Intelligence 86 (1996) 195-244

Artificial Intelligence

Mechanism design for automated negotiation, and its application to task oriented domains ☆

Gilad Zlotkin[a,1], Jeffrey S. Rosenschein[b,*]

[a] *Center for Coordination Science, Sloan School of Management, MIT, 1 Amherst St., E40-179, Cambridge, MA 02139, USA*

[b] *Institute of Computer Science, Hebrew University, Givat Ram, Jerusalem, Israel*

Received November 1993

Abstract

As distributed systems of computers play an increasingly important role in society, it will be necessary to consider ways in which these machines can be made to interact effectively. Especially when the interacting machines have been independently designed, it is essential that the *interaction environment* be conducive to the aims of their designers. These designers might, for example, wish their machines to behave efficiently, and with a minimum of overhead required by the coordination mechanism itself. The rules of interaction should satisfy these needs, and others. Formal tools and analysis can help in the appropriate design of these rules.

We here consider how concepts from game theory can provide standards to be used in the design of appropriate negotiation and interaction environments. This design is highly sensitive to the domain in which the interaction is taking place. Different interaction mechanisms are suitable for different domains, if attributes like efficiency and stability are to be maintained.

We present a general theory that captures the relationship between certain domains and negotiation mechanisms. The analysis makes it possible to categorize precisely the kinds of domains in which agents find themselves, and to use the category to choose appropriate negotiation mechanisms. The theory presented here both generalizes previous results, and allows agent designers to characterize new domains accurately. The analysis thus serves as a critical step in using the theory of negotiation in real-world applications.

We show that in certain task oriented domains, there exist distributed consensus mechanisms with simple and stable strategies that lead to efficient outcomes, even when agents have incomplete information about their environment. We also present additional novel results, in particular that

☆ This is a revised and expanded version of the papers "A domain theory for task oriented negotiation" and "Consenting agents: negotiation mechanisms for multi-agent systems" that appeared in Proceedings IJCAI-93, Chambery, France, pp. 416-422 and pp. 792-799 respectively.

* Corresponding author. E-mail: jeff@cs.huji.ac.il.

[1] E-mail: gilad@mit.edu.

in concave domains using all-or-nothing deals, no lying by an agent can be beneficial, and that in subadditive domains, there often exist beneficial decoy lies that do not require full information regarding the other agent's goals.

1. Machines controlling and sharing resources

Computers are making more and more decisions in a relatively autonomous fashion. Telecommunication networks are controlled by computers that decide on the routing of telephone calls and data packets. Electrical grids have decisions made by computer regarding how their loads will be balanced at times of peak demand. Similarly, research is being done on how computers can react to, and control, automotive and airplane traffic in real time.

Some of the decisions that these computers are generating are made in concert with other machines. Often, this inter-machine consultation is crucial to the task at hand. For example, with personal digital assistants (PDAs), the individual's palmtop computer will be expected to coordinate schedules with others' PDAs (e.g., my software agent determines whether my car has been fixed on time at the garage; if not, it contacts the taxi company, reschedules my order for a cab, and updates my day's schedule). No scheduling will take place without inter-machine communication. Rarely will it take place without the resolution of inter-machine conflict (because the humans that these machines represent have conflicting goals).

Similarly, the concept of intelligent databases relies on sophisticated interactions among autonomous software agents. A user's request for a piece of information may require collecting and synthesizing information from several distributed databases. Machines need to formulate the necessary collection of requests, arrange access to the data (which may be partially restricted), and cooperate to get the information where it is needed.

Even when a computer's tasks do not *have* to involve other machines, it may be beneficial to involve them. Sometimes, for example, we find automated systems controlling resources (like the telecommunications network mentioned above). It is often to the benefit of separate resource-controlling systems to share their resources (e.g., fiber optic lines, short and long term storage, switching nodes) with one another.

All of this inter-machine coordination will be taking place within some kind of *interaction environment*. There will inevitably be "protocols" for how machines deal with one another. What concerns us here are not the details of how to stuff information into a packet on the network; it's not even the higher-level issue of how agents will communicate with one another (in a common language, or perhaps using translation filters). Rather, once we assume that agents *can* communicate and understand one another, how will they come to agreements? These "interaction rules" will establish the basis for inter-machine negotiation, agreement, coordination, and cooperation.

If the inter-machine protocols are primitive and incapable of capturing the subtleties of cooperative opportunities, the machines will act inefficiently. They will make the wrong decisions. The people who depend on those decisions will suffer.

1.1. Overview of the article

This article is organized into three parts:

(1) In the first part (Sections 1.2-9), we discuss the general approach of our framework for automated negotiation, along with its motivation and related research. We also give a very brief and informal overview of our results.

(2) In the second part (Sections 10-14), we present the formal details of our application of the framework to task oriented domains. This second part also includes a more formal discussion of some of the main issues raised in the first part of the article.

(3) In Appendix A, we present the proofs to all theorems contained in the second part of the article.

1.2. Heterogeneous, self-motivated agents

In the field of distributed artificial intelligence (DAI), researchers explore methods that enable the coherent (e.g., coordinated, efficient) interaction of computers in distributed systems. One of the major distinctions in DAI is between research in distributed problem solving (DPS) [4, 8, 13, 63], in which the distributed system has been centrally designed, and multiagent (MA) systems [18, 38, 40, 68], in which the distributed system is made up of independently designed agents. In DPS, there is some global task that the system is performing, and there exists (at least implicitly) a global notion of utility that can constrain or direct agent activity. In DPS, there is assumed to be a single body who is able, at design time, to directly influence the preferences of all agents in the system.

In multiagent systems, there is assumed to be *no single body* who is able, at design time, to directly influence the preferences of all agents in the system. The agents' preferences arise from distinct designers. In MA systems, each agent is concerned only with its own goals (though different agents' goals may overlap), and there is no global notion of utility. The MA system agent, concerned with its own welfare (i.e., the welfare of its designer [12]), acts accordingly to increase that welfare.

The distinction between distributed problem solving and multiagent systems should really be seen primarily as a distinction between research agendas.[2] Certainly it will not always be obvious to an outside observer whether a given distributed system falls into one category or the other. A single designer may have built his agents to act competitively, believing it improves overall system efficiency. Similarly, individually motivated agents might be seen sharing information and helping one another, because they have determined that it is in their own best interests to act that way. However, the research questions asked by a researcher in DPS are sometimes distinct from those asked by an MAS researcher (despite a good deal of overlap). In particular, if a DPS researcher can show that acting in a particular way is good for the system as a whole,

[2] In fact, the use of these terms is itself undergoing a change. "Multiagent systems" is now sometimes used to refer to the entire field of distributed artificial intelligence. For more of a discussion about the DPS/MAS distinction, see [16].

he can *impose* this behavior on all the agents in the system at design time. For the MAS researcher, such an alternative is unavailable. At best, he might be able to design aspects of the environment that motivate all the (selfish) agents to act in a certain way. This need for *indirect* incentives is one element that distinguishes MAS research from DPS research.

The approach of MA system research is particularly appropriate for the kinds of scenarios mentioned above. When AT&T and MCI computers communicate with the purpose of load balancing their message traffic, each is concerned with its own company's welfare. Any interaction environment must take into account that each of these software agents, in coming to an agreement, will be primarily concerned with its own increased benefit from that agreement. We are not looking for benevolent or altruistic behavior from these machines. Similarly, these systems of interacting machines tend to be "open" [26], in the sense that the system composition is not fixed. With PDAs, for example, new agents (and even new types of agents) will constantly be entering the environment. My PDA, to be effective in negotiation and coordination, must be able to deal with these open, dynamic, configurations of agents. Research in multiagent systems is thus the appropriate model with which to analyze these independent software agents and their interactions.

2. Related work in distributed artificial intelligence

There have been several streams of research in DAI that have approached the problem of multiagent coordination in different ways. We here briefly review some of this work, categorizing it in the general areas of multiagent planning, negotiation, social laws, and economic approaches.

2.1. *Multiagent planning*

One focus of DAI research has been that of "planning for multiple agents", which considers issues inherent in centrally directed multiagent execution. Smith's contract net [63,64] falls into this category, as does other DAI work such as [35,51,55]. A second focus for research has been "distributed planning", where multiple agents all participate in coordinating and deciding upon their actions [10,15,18,53,56,58,78].

The question of whether the group activity is fashioned centrally or in a distributed manner is only one axis of comparison. Another important issue that distinguishes various DAI research efforts is whether the goals themselves need to be adjusted, that is, whether there may be any fundamental conflicts among different agents' goals. Thus, for example, Georgeff's early work on multiagent planning assumed that there was no basic conflict among agent goals, and that coordination was all that was necessary to guarantee success [28,29,66]. Similarly, planning in the context of Lesser, Corkill, and Durfee's research [11] often involves coordination of activities (e.g., sensor network computations) among agents that have no inherent conflict with one another (though surface conflict may exist). "Planning" here means avoidance of redundant or distracting activity, efficient exploration of the search space, etc.

Another important issue is the relationship that agents have to one another, e.g., the degree to which they are willing to compromise their goals for one another (assuming that such compromise is necessary). *Benevolent agents* [58] are those that, by design, are willing to accommodate one another; they have been built to be cooperative, to share information, and to coordinate in pursuit of some (at least implicit) notion of global utility. In contrast, multiagent system agents will cooperate only when it is in their best interests to do so. Still another potential relationship among agents is a modified master-slave relationship, called a "supervisor-supervised" relationship, where non-absolute control is exerted by one agent over another [19, 20].

The synthesis, synchronization, or adjustment process for multiple agent plans thus constitute some of the (varied) foci of DAI planning research. Synchronization through conflict avoidance [28,29,66], distribution of a single-agent planner among multiple agents [9], the use of a centralized multiagent planner [55], and the use of consensus mechanisms for aggregating subplans produced by multiple agents [23], have all been explored. Other recent work includes [17,24,34,36,40,69-71].

2.2. *Axiomatic approaches to group activity*

There exists a large and growing body of work within artificial intelligence that attempts to capture notions of rational behavior through logical axiomatization [6, 7,30,31,37,46,47,54]. The approach usually centers on a formalized model of the agent's beliefs, desires, and intentions (the so-called "BDI model"). The purpose of the formal model is to precisely characterize what constitutes rational behavior, with the intent of imposing such rational behavior on an automated agent. The formal axioms might be used at run-time to directly constrain an agent's decision process, or (more likely) they could be used at compile-time to produce a more efficient executable module.

The focus of this research, coming as it does from a single-agent artificial intelligence perspective, is on the architecture of a single automated agent. For example, Cohen and Levesque [5,6] have explored the relationship between choice, commitment, and intention—an agent should commit itself to certain plans of action, and remain loyal to these plans as long as it is appropriate (for example, when the agent discovers a plan is infeasible, the plan should be dropped).

Even when looking at multiagent systems, these researchers have examined how a member of a group should be designed—again, looking at how to design an individual agent so that it is a productive group member. For example, in [36] axioms are proposed that cause an agent, when he discovers that he will fail to fulfill his role in a joint plan, to notify the other members of his group. Axiomatizations, however, might need to deal with how *groups* of agents could have a *joint* commitment to accomplishing some goal [7], or how each agent can make interpersonal commitments without the use of such notions [31]. Another use for the BDI abstractions is to allow one agent to reason about other agents, and relativize one's intentions in terms of beliefs about other agents' intentions or beliefs.

2.3. *Social laws for multiple agents*

Various researchers in distributed artificial intelligence have suggested that it would be worthwhile to isolate "aspects of cooperative behavior", general rules that would cause agents to act in ways conducive to cooperation. The hypothesis is that when agents act in certain ways (e.g., share information, act in predictable ways, defer globally constraining choices), it will be easier for them to carry out effective joint action [3,45,65].

Moses, Shoham, and Tennenholtz [48,49,61,62], for example, have suggested applying the *society metaphor* to artificial systems to improve the performance of the agents operating in this society. The issues that are to be dealt with are synchronization, coordination of the agents' activities, cooperative ways to achieve tasks, and how safety and fairness constraints on the system can be guaranteed. They propose coordinating agent activity to avoid conflicts; the system will be structured so that agents will not arrive at potential conflict situations.

Thus these social laws are seen as a method to avoid the necessity for costly coordination techniques, like planning or negotiation. With agents following the appropriate social laws, the need for run-time coordination will be reduced. This is important, because although agent designers may be willing to invest a large amount of effort at design time in building effective distributed systems, it is important that the run-time overhead be as low as possible.

There is a similarity between this use of pre-compiled, highly structured social laws, and our development of pre-defined interaction protocols. However, the social laws approach assumes that the designer of the laws has full control over the agents; agents are assumed to follow the social laws simply because they were designed to, and not because they individually benefit from the social laws. Obeying the social laws may not be "stable"; assuming that everyone else obeys the laws, an agent might do better by breaking them. Our approach is concerned with social conventions that are stable, which will be suitable for individually motivated agents.

2.4. *Economic approaches*

There have been several attempts to consider market mechanisms as a way of efficiently allocating resources in a distributed system. Among the AI work is that of Smith's contract net [63,64], Malone's enterprise system [44], and Wellman's WALRAS system [72,73].

The contract net is a high-level communication protocol for a distributed problem solving system. It enables the distribution of the tasks among the nodes that operate in the system. A contract between two nodes is established so that tasks can be executed; each node in the net can act as a *manager* and/or as a *contractor*. A task that has been assigned to a node can be further decomposed by the contractor. A contract is established by a bidding scheme that includes the announcement of the task by the manager, and bids sent in by the potential contractors.

Enterprise [44] is a system that was built using a variation of the contract net protocol. The *distributed scheduling protocol* locates the best available machine to perform a task. This protocol is similar to the contract net, but makes use of a more well-defined assignment criteria.

Another system that takes an economic approach in solving a distributed problem through the use of a price mechanism has been explored by Wellman in [72]. Wellman uses the consumer/producer metaphor to establish a market pricing-based mechanism for task redistribution that ensures stability and efficiency. All agents act as both consumers and producers. Each distinct good has an *auction* associated with it, and agents can get the good by submitting bids in the auction for that good. The system developed by Wellman, WALRAS, computes for each market the equilibrium price.

2.5. *Negotiation*

Negotiation has been a subject of central interest in DAI. The word has been used in a variety of ways, though in general it refers to communication processes that further coordination [8,40,41,63]. These negotiating procedures have included the exchange of partial global plans [13], the communication of information intended to alter other agents' goals [67,68], and the use of incremental suggestions leading to joint plans of action [38].

Interagent collaboration in distributed problem solving systems has been explored in the ongoing research of Lesser, Durfee, and colleagues. Much of this work has focused on the implementation and analysis of data fusion experiments, where systems of distributed sensors absorb and interpret data, ultimately arriving at a group conclusion [11,14,42]. Agents exchange partial solutions at various levels of detail to construct global solutions; much of the work has examined effective strategies for communication of data and hypotheses among agents, and in particular the kinds of relationships among nodes that can aid effective group analysis. For example, different organizations, and different methods for focusing node activity, can help the system as a whole be far more efficient.

Sycara has examined a model of negotiation that combines case-based reasoning and optimization of multi-attribute utilities. In particular, while in our research we assume that agents' goals are fixed during the negotiation, Sycara is specifically interested in how agents can influence one another to *change* their goals through a process of negotiation (information transfer, etc.).

Kraus and her colleagues have explored negotiation where the negotiation time itself is an issue [38,39]. Agents may lose value from a negotiation that drags out too long, and different agents are asymmetric with regard to the cost of negotiation time. Agents' attitudes towards negotiation time directly influences the kinds of agreements they will reach. Interestingly, however, those agreements can be reached without delay. There is an avoidable inefficiency in delaying agreement.

Gasser [27] has explored the social aspects of agent knowledge and action in multiagent systems ("communities of programs"). Social mechanisms can dynamically emerge; communities of programs can generate, modify, and codify their own local languages of interaction. Gasser's approach may be most effective when agents are interacting in unstructured domains, or in domains where their structure is continuously changing. The research we present, on the other hand, exploits a pre-designed social layer for multiagent systems.

Ephrati and Rosenschein [18,21,22] used the Clarke Tax voting procedure as an n-agent consensus mechanism, in essence to *avoid* the need for classical negotiation. The mechanism assumes the ability to transfer utility explicitly. The Clarke Tax technique in fact assumes (and requires) that agents are able to transfer utility out of the system (taxes that are paid by the agents). The utility that is transferred out of the system is actually wasted, and reduces the efficiency of the overall mechanism. This, however, is the price paid to ensure stability. The work we present below does not assume the explicit transfer of utility (though implicit transfer is possible, to a certain extent). Also, the negotiation mechanism ensures stability without the inefficiency of transfering utility out of the system.

2.5.1. Relationship to our previous work on negotiation

In previous work [75,77-80], we considered various negotiation protocols in different domains, and examined their properties. Agents were assumed to have a goal that specified a set of acceptable final states. These agents then entered into an iterative process of offers and counter-offers, exploring the possibility of achieving their goals at lower cost, and/or resolving conflicts between their goals.

The procedure for making offers was formalized in a *negotiation mechanism*; it also specified the form that the agents' offers could take (deal types). A deal between agents was generally a joint plan. The plan was "joint" in the sense that the agents might probabilistically share the load, compromise over which agent does which actions, or even compromise over which agent gets which parts of its goal satisfied.

The interaction between agents occurs in two consecutive stages. First the agents negotiate, then they execute the entire joint plan that has been agreed upon. No divergence from the agreed deal is allowed (i.e., after the deal is publically agreed upon, appropriate behavior can be externally enforced). The sharp separation of stages has consequences, in that it rules out certain negotiation tactics that might be used in an interleaved process.

At each step, both agents simultaneously offer a deal. Our "monotonic concession protocol" (introduced in [75]) specified that at no point could an agent demand more than it did previously—in other words, each offer either repeated the previous offer or conceded by demanding less. The negotiation ended in one of two ways:

- *conflict*: if neither agent makes a concession at some step, they have by default agreed on the (domain dependent) "conflict deal";
- *agreement*: if at some step an agent A_1, for example, offers agent A_2 more than A_2 himself asks for, they agree on A_1's offer, and if both agents overshoot the other's demands, then a coin toss breaks the symmetry.

The result of these rules is that agents cannot backtrack, nor can they both simultaneously "stand still" in the negotiation more than once (since the first such occurrence causes them to reach a conflict). Thus the negotiation process is strongly monotonic and ensures convergence to a deal.

Deal types explored in our previous work included *pure deals*, *all-or-nothing deals*, *mixed deals*, *joint plans*, *mixed joint plans*, *semi-cooperative deals*, and *multi-plan deals*. Each of these types of agreement proved suitable for solving different kinds of interactions. For example, semi-cooperative deals proved capable of resolving true conflicts between agents, whereas mixed deals did not. Similarly, multi-plan deals are capable of capturing goal relaxation as part of an agreement.

It was also shown that certain other properties were true of some deal types but not of others. In particular, different agent strategies were appropriate ("rational") for different deal types and domains. Agents were shown to have no incentive to lie when certain deal types were used in certain domains, but did have an incentive to lie with other deal type/domain combinations.

The examination of this relationship between the negotiation mechanism and the domain made use of two prototypical examples: the postmen domain (introduced in [75]), and the slotted blocks world (presented in [77]). It was clear that these two domains exemplified general classes of multiagent interactions (e.g., the postmen domain was inherently cooperative, the slotted blocks world not). It was, however, not clear what attributes of the domains made certain negotiation mechanisms appropriate for them. Nor was it clear how other domains might compare with these prototypes. When presented with a new domain (such as agents querying a common database), which previous results were applicable, and which weren't? The research lacked a general theory explaining the relationship between domains and negotiation mechanisms.

In this article, we present the beginnings of such a general theory. The analysis makes it possible both to understand previous results in the postmen domain more generally, and to characterize new domains accurately (i.e., what negotiation mechanisms are appropriate). The analysis thus serves as a critical step in using the theory of negotiation in real-world applications.

3. The aim of the research

The purpose of the research described in this article is to consider how we might build machines that are capable of making constructive agreements. We want our machines to interact flexibly. We want them to represent our interests, and compromise when that is to our advantage. We may want them to be secretive at times, not revealing all their information, and we most likely want them to recognize duplicity on the part of others, when possible. In short, we want our agents to faithfully act as our surrogates in encounters with other agents.

3.1. Social engineering for machines

When humans interact, they do not do so in a vacuum. There are social conventions and laws that constrain their behavior; the purpose of social conventions and laws is to do exactly that. A tax levied on a company that pollutes the air is intended as a disincentive to a certain kind of behavior. Positive publicity showered on a philanthropic company provides it with benefit for its behavior. One can think of a complicated system of laws and conventions as a kind of social engineering, intended to produce certain behavior among people.

We are interested in social engineering for machines. We want to understand the kinds of negotiation protocols, and punitive and incentive mechanisms, that would motivate individual designers to build machines that act in ways that all those designers find beneficial. As mentioned above, the development of "social laws" has parallels with the work of [62]. There, however, the social laws are for centrally designed systems of agents (DPS), and will not necessarily make sense for independently designed agents. For example, a rule might encourage efficient behavior if everyone followed it, but if any single agent could benefit more by *not* following the rule, the system as a whole will not be stable. Since each of our agents will do what is necessary to maximize its benefit, stability is a critical issue—we need rules that agents will independently find in their best interests to follow. We will return to this issue of stability below.

3.2. The setting of standards

The scenario we consider is as follows. Imagine representatives of various companies (agent designers) coming together to agree on interaction protocols for their automated agents. Given a particular domain (such as balancing telecommunications traffic among wide area networks, or meeting scheduling), they are presented with various interaction mechanisms, and shown that each mechanism has certain provable properties. For example, one mechanism might arrive at guaranteed globally optimal solutions, but at the cost of one agent possibly doing very badly. Another mechanism might ensure that the gap between agents' benefits are minimized, but at the cost of everyone doing a little worse. Moreover, it is shown to these company representatives that Protocol A is immune to deception: it will be in no one's interest to design a cheating agent that deviates from the protocol in any way (e.g., by reporting higher, or lower, network traffic than is actually present). The representatives consider the various options, and decide among themselves which protocol to build into their agents. The meeting adjourns, agents are built, and beneficial agreements are reached among them.

It turns out that the attributes of a given mechanism are highly sensitive to the domain in which the agents are operating. The rules of interaction that might be appropriate in one domain might be quite inappropriate in another. When those company representatives sit down at the meeting, they need to be told "In this domain, Protocol A has properties 1, 2, and 3, and is immune to deception. Protocol B has properties 2, 4, and 5, and is not immune to deception." Our research explores the space of possibilities, analyzing negotiation mechanisms in different domains. When the designers of automated agents meet, this is the kind of information they will need. The alternative to having this analysis is to wander in the dark, and to build negotiation modules without understanding their properties. Will they result in good deals? Could our machines do better? Will someone build a deceptive agent that takes advantage of mine? Should I, myself, design my agent to be secretive or deceptive? Will this further my own goals? Our research is intended to answer these kinds of questions.

The builders of complex distributed systems, like interconnected networks, shared databases, assembly line monitoring and manufacturing, and distributed processing, can broaden the range of tools that they bring to bear on issues of interagent coordination.

Existing techniques generally rely on the goodwill of individual agents, and don't take into account complex interactions of competing goals. New tools can be applied to the high-level design of heterogeneous, distributed systems through the creation of appropriate negotiation protocols.

4. Protocol design

How can machines decide how to share resources, or which machine will give way while the other proceeds? Negotiation and compromise are necessary, but how do we build our machines to do these things? How can the designers of these separate machines decide on techniques for agreement that enable mutually beneficial behavior? What techniques are appropriate? Can we make definite statements about the techniques' properties?

The way we have begun to address these questions is to synthesize ideas from artificial intelligence (e.g., the concept of a reasoning, rational computer) with the tools of game theory (e.g., the study of rational behavior in an encounter between self-interested agents). Assuming that automated agents, built by separate, self-interested designers, will interact, we are interested in designing *protocols* for specific domains that will get those agents to interact in useful ways.

The word "protocol" means different things to different people. As used to describe networks, a protocol is the structure of messages that allow computers to pass information to one another. When we use the word protocol, we mean the rules by which agents will come to agreements. It specifies the kinds of deals they can make, as well as the sequence of offers and counter-offers that are allowed. These are high-level protocols, dealing not with the mechanisms of communication but with its content. Protocols are intimately connected with *domains*, by which we mean the environment in which our agents operate. Automated agents that control telecommunications networks are operating in a different domain (in a formal sense) than robots moving boxes. Much of our research is focused on the relationship between different kinds of domains, and the protocols that are suitable for each.

Given a protocol, we need to consider what agent *strategy* is appropriate. A strategy is the way an agent behaves in an interaction. The protocol specifies the rules of the interaction, but the exact deals that an agent proposes is a result of the strategy that his designer has put into him. As an analogy, a protocol is like the rules governing movement of pieces in the game of chess. A strategy is the way in which a chess player decides on his next move.

4.1. The game theory/automated agent match

Game theory is the right tool in the right place for the design of automated interactions. Game theory tools have been primarily applied to analyzing human behavior, but in certain ways they are inappropriate: humans are not always rational beings, nor do they necessarily have consistent preferences over alternatives. Automated societies, on the other hand, are particularly amenable to formal analysis and design. Automated agents can exhibit predictability, consistency, narrowness of purpose (e.g., no emotions, no humor, no fears, clearly defined and consistent risk attitude), and an explicit measurement of utility (where this can have an operative meaning inside the program controlling the agent).

Even the notion of "strategy" (a specification of what to do in every alternative during an interaction), a classic game theory term, takes on a clear and unambiguous meaning when it becomes simply a program put into a computer. The notion that a human would choose a fixed strategy before an interaction, and follow it without alteration, leads to unintuitive results for a person. Moreover, it seems to be more a formal construct than a realistic requirement—do humans consider *every* alternative ahead of time and decide what to do? On the other hand, the notion that a computer is programmed with a fixed strategy before an interaction, and follows it without alteration, is a simple description of the current reality.

Of course, neither humans nor computer programs are ideal game theory agents. Most importantly, they are not capable of unlimited reasoning power, as game theory often assumes. Nevertheless, it seems that in certain ways automated agents are closer to the game theory idealization of an agent than humans are. The work described here, the design of interaction environments for machines, is most closely related to the field of mechanism design in game theory [25].

4.2. Comparison with game theory

4.2.1. Bargaining theory

The approach taken in this article is strongly based on previous work in game theory, primarily on what is known as "Nash's bargaining problem" [43] or "Nash's model of bargaining" [60].

Classic game theory [32,43,50,60,74] talks about agents reaching "deals", which are defined as vectors of utilities (one for each agent). A bargaining game can end up in some possible outcome (i.e., a "deal"). Each player has a full preference order over the set of possible outcomes; this preference order is expressed by its utility function. For each deal, there is a utility vector which is the list of the utilities of this deal for every participant. There is a special utility vector called "conflict" (or sometimes the "status quo point") which is the utility each player assigns to a conflict (lack of final agreement). Game theory assumes that the set of possible deals is a simplex, i.e., if u_1 and u_2 are two utility vectors for two possible deals, then every vector on the line connecting u_1 to u_2 is a utility vector of some possible deal (meaning that the domain of deals is continuous). Classic game theory deals with the following question: given a set of utility vectors (a simplex), what will be the utility vector that the players will agree on (under particular assumptions)? In other words, classic bargaining theory is focused on prediction of outcomes, under certain assumptions about the players and the outcomes themselves.

Nash [50] showed that under some rational behavior and symmetry assumptions, players will reach an agreement on a deal that will be *individual rational, pareto optimal*, and will maximize the product of the players' utility (see Section 13 for a more complete discussion).

Zeuthen [74] considered the two-player bargaining problem as a one-player decision process (deciding whether to concede or not), under the assumption that if none of the players concede at a particular step they will reach a conflict. Zeuthen evaluated how much risk each player would be willing to take when he decides not to concede (and thereby risks conflict). The player who is least willing to risk will be the one who will make the next concession.

Harsanyi [32] showed that the two approaches are equivalent in the sense that two players using Zeuthen's criteria will reach the Nash solution.

The above approaches furnish us with tools for our own design and evaluation of negotiation mechanisms, which is our primary concern. Game theorists are usually concerned with how games will be played, from both a descriptive and normative point of view. Game solutions consist of strategies in equilibrium; if somehow a social behavior reaches an equilibrium, no agent has any incentive to diverge from that equilibrium behavior. That equilibrium is considered to be a solution to the game. There may be one or more (or no) strategies in equilibrium, and there are also different notions of equilibrium (e.g., Nash [50], perfect equilibrium, dominant strategy equilibrium).

There are also groups of game theorists who consider the problem of how to design games that have certain attributes. It is this area of mechanism design that is closest to our own concerns for automated agents.

4.2.2. Mechanism design and implementation theory

Mechanism design is also know in the game theory literature as the *implementation* problem. The *implementation question* [2,25] asks whether there is a *mechanism*, or *game form*, with a distinguishable equilibrium point (dominant strategy, or strong, or merely Nash) such that each social profile (i.e., group behavior) is associated, when the players follow their equilibrium strategies, with the desired outcome.

In other words, there are assumed to be a group of agents, each with its own utility function and preferences over possible social outcomes. There is also a social welfare function that rates all those possible social outcomes (e.g., a socially efficient agreement may be rated higher than a non-efficient one). The question is then, can one design a game such that it has a unique solution (equilibrium strategies), and such that when each individual agent behaves according to this equilibrium strategy, the social behavior will maximize the social welfare function. If such a game can be designed, then it is said that the game implements the social welfare function.

As an example of a social welfare function, consider minimization of pollution. While everyone may be interested in lowering pollution, everyone is interested in others bearing the associated costs. A mechanism to implement this social welfare function might include, for example, taxes on polluting industries and tax credits given for the purchase of electric cars. This is precisely the kind of mechanism that would cause agents, following an equilibrium strategy, to minimize pollution.

4.2.3. Incentive compatibility

The designer of a mechanism has to deal with every configuration of agent utility functions (which is precisely why he is designing a mechanism, or "game form", and not a specific game). The agents themselves may or may not have complete information about one another's utility functions, and thus may or may not know which concrete game they are playing. This complicates the problem of mechanism design. Usually, there is an assumption that the agents have certain limitations on the form of their utility functions. Thus there exists a known set of all possible utility functions. Each agent can then be assigned a "type" based on which of those utility functions it is currently using.

A mechanism is called a *direct* mechanism if the agents are asked straight out what their type is. Then, based on the agents' declared types, the mechanism generates some outcome. If the agents are not asked their type, the mechanism is called an *indirect* mechanism. The *revelation* principle states that whatever can be done with an indirect mechanism can also be done with a direct mechanism. In other words, any social function that is implementable by an indirect mechanism can also be implemented by a direct mechanism, where agents will have an incentive to declare their true type. This is called an incentive compatible mechanism.

The advantage of a direct mechanism where the agents have an incentive to declare their true type over an indirect mechanism is the simplicity of the agents' strategies. However, the simplicity on the part of the agents may be offset by the need for a complicated mechanism. While we prefer a direct mechanism, it is also important to us that it be a simple one.

One can look at the work described in this article as a kind of mechanism design, where the social welfare function that is being implemented is "efficiency", i.e., the sum of agents' utilities. Requiring that the sum be maximized *a priori* rules out many social behaviors, but still may allow multiple sum maximization behaviors. In other words, even when the sum is being maximized, the way in which utility is divided among agents may differ. Since each agent wants a bigger share of this group utility, we have a typical negotiation scenario. Either the agents agree on a division, or they reach a conflict.

Since we are talking about automated agents, we depend on their designers to reach a consensus about the mechanism and its associated strategies. Therefore, our mechanism design is simpler than that of game theory. In contrast to classic mechanism design, we are not concerned with the uniqueness of the equilibrium strategies. What is important to us is the *existence* of an equilibrium strategy that maximizes the social welfare function. We are satisfied with the existence of even one such maximizing equilibrium social behavior, even though there may be other social behaviors in equilibrium that do not maximize the social welfare function. The designers of our automated agents can choose a mechanism that has many equilibrium points (some of which maximize the social welfare function and some of which don't), but then coordinate themselves (by jointly choosing a strategy) so that their agents reach a particular equilibrium point that maximizes the social welfare function. By relaxing our requirement of the mechanism, we may discover social welfare functions that can be implemented using our approach, but cannot be implemented using game theory's (stricter) approach.

5. Attributes of standards

What are the attributes that might interest those company representatives when they meet to discuss the interaction environment for their machines? This set of attributes, and their relative importance, will ultimately affect their choice of interaction rules.

We have considered several attributes that might be important to system designers.

(1) *Efficiency*: The agents should not squander resources when they come to an agreement; there should not be wasted utility when an agreement is reached. For example, it makes sense for the agreements to satisfy the requirement of pareto optimality (no agent could derive more from a different agreement, without some other agent deriving less from that alternate agreement). Another consideration might be global optimality, which is achieved when the sum of the agents' benefits are maximized. Neither kind of optimality necessarily implies the other. Since we are speaking about self-motivated agents (who care about their own utilities, not the sum of system-wide utilities—no agent in general would be willing to accept lower utility just to increase the system's sum), pareto optimality plays a primary role in our efficiency evaluation. Among pareto optimal solutions, however, we might also consider as a secondary criterion those solutions that increase the sum of system-wide utilities.

(2) *Stability*: No designer should have an incentive to deviate from agreed-upon strategies. The strategy that agents are programmed with can be proposed as part of the interaction environment design. Once these strategies have been proposed, however, we do not want individual designers (e.g., companies) to have an incentive to go back and build their agents with different, manipulative, strategies.

(3) *Simplicity*: It will be desirable for the overall interaction environment to make low computational demands on the agents, and to require little communication overhead. This is related both to efficiency and to stability: if the interaction mechanism is simple, it increases efficiency of the system, with fewer resources used up in carrying out the negotiation itself. Similarly, with stable mechanisms, few resources need to be spent on outguessing your opponent, or trying to discover his optimal choices. The optimal behavior has been publicly revealed, and there is nothing better to do than just carry it out.

(4) *Distribution*: Preferably, the interaction rules will not require a central decision maker, for all the obvious reasons. We do not want our distributed system to have a performance bottle-neck, nor collapse due to the single failure of a special node.

(5) *Symmetry*: No designer wants the negotiation process to be arbitrarily biased against his agent. Thus, no mechanism should treat agents differently because of inappropriate criteria. Exactly what constitutes inappropriate criteria depends on the specific domain. While the mechanism may be asymmetric, for example because one agent has many more tasks to carry out than the other, it should maintain strict impartiality when irrelevant aspects of the agents differ (e.g., the agents' manufacturers).

These attributes need not be universally accepted. In fact, there will sometimes be trade-offs between one attribute and another (for example, efficiency and stability are sometimes in conflict with one another [83]). But our protocols are designed, for specific classes of domains, so that they satisfy some or all of these attributes. Ultimately, these are the kinds of criteria that rate the acceptability of one interaction mechanism over another.

As one example, the attribute of stability assumes particular importance when we consider open systems, where new agents are constantly entering and leaving the community of interacting machines. Here, we might want to maintain stability in the face of new agents who bring with them new goals and potentially new strategies as well. If the mechanism is "self-perpetuating", in that it is not only to the benefit of society as a whole to follow the rules, but also to the benefit of each individual member, then the social behavior remains stable even when the society's members change dynamically. When the interaction rules create an environment in which a particular strategy is optimal, beneficial social behavior is resistant to outside invasion [1].

6. Domain theory

We have several times alluded to the connection between protocols and domains—for a given class of interactions, some protocols might be suitable while others are not. We have found it useful to categorize domains into a three-tier hierarchy of task oriented domains, state oriented domains, and worth oriented domains. This hierarchy is by no means complete, but does cover a large proportion of the kinds of real-world interactions in which we are interested. This article is focused on task oriented domains. For treatments of state oriented domains, see [76–78, 82–84]; for treatments of worth oriented domains, see [59, 79, 85].

6.1. *Task oriented domains*

These are domains in which an agent's activity can be defined in terms of a set of tasks that it has to achieve. These tasks can be carried out without concern about interference from other agents; all the resources necessary to accomplish the tasks are available to the agent. On the other hand, it is possible that agents can reach agreements where they redistribute some tasks, to everyone's benefit (for example, if one agent is doing some task, he may, at little or no cost, be able to do another agent's task). The domains are inherently cooperative. Negotiation is aimed at discovering mutually beneficial task redistribution.

The key issue here is the notion of *task*, an indivisible job that needs to be carried out. Of course, what constitutes a task will be specific to the domain. Many kinds of activity, however, can be conceived of in this way, as the execution of indivisible tasks. For example, imagine that you have three children, each of whom needs to be delivered to a different school each morning. Your neighbor has four children, and also needs to take them to school. Delivery of each child can be modeled as an indivisible task. Although both you and your neighbor might be interested in setting up a carpool, there is no doubt that you will be able to achieve your tasks by yourself, if necessary. The worst that can happen is that you and your neighbor won't come to an agreement about setting up a carpool, in which case you are no worse off than if you were alone. You can only benefit (or do no worse) from your neighbor's existence.

Assume, though, that one of my children and one of my neighbor's children both go to the same school (that is, the cost of carrying out these two deliveries, or two tasks, is the same as the cost of carrying out one of them). It obviously makes sense for both children to be taken together, and only my neighbor or I will need to make the trip to carry out both tasks.

What kinds of agreements might we reach? We might decide that I will take the children on even days each month, and my neighbor will take them on odd days; perhaps, if there are other children involved, we might have my neighbor always take those two specific children, while I am responsible for the rest of the children (his and mine). Another possibility would be for us to flip a coin every morning to decide who will take the children. An important issue, beyond *what* deals can be reached, is *how* a specific deal will be agreed upon (see Section 7.2 below).

Consider, as further examples, the postmen domain, the database domain, and the fax domain (these domains are described in more detail, and more formally, below). In the postmen domain, each agent is given a set of letters to deliver to various nodes on a graph; starting and ending at the post office, the agents are to traverse the graph and make their deliveries. There is no cost associated with carrying letters (they can carry any number), but there is a cost associated with graph traversal. The agents are interested in making short trips. Agents can reach agreements to carry one another's letters, and save on their travel.

The database domain similarly assigns to each agent a set of tasks, and allows for the possibility of beneficial task redistribution. Here, each agent is given a query that it will make against a common database (to extract a set of records). A query, in turn, may be composed of subqueries (i.e., the agent's tasks). For example, one agent may want the records of "All female employees making over $50,000 a year", while another agent may want the records of "All female employees with more than three children". Both agents share a subtask, the query that involves extracting the records of all female employees (prior to extracting a subset of those records). By having only one agent get the female employee records, another agent can lower its cost.

The third example is the fax domain. It appears very similar to the postmen domain, but is subtly different. In the fax domain, each agent is given a set of faxes to send to different locations around the world (each fax is a task). The only cost is to establish a connection. Once the connection is made, an unlimited number of faxes can be sent. Of course, if two agents both have faxes to send to Paris and to London, they may redistribute their faxes, with one sending all the faxes to Paris and the other sending all the faxes to London.

Despite the seemingly minor differences in these domains, the attributes of suitable protocols are very different for each, as we will see below.

6.2. State oriented domains

The state oriented domain (SOD) is the type of domain with which most AI research has dealt. The blocks world, for example, is a classic state oriented domain. SODs are a superset of TODs (i.e., every TOD can be cast in the form of an SOD).

In an SOD, each agent is concerned with moving the world from an initial state into one of a set of goal states. There is, of course, the possibility of real conflict here. Because of, for example, competition over resources, agents might have fundamentally different goals. There may be no goal states that satisfy all agents. At other times, there may exist goal states that satisfy all agents, but that are expensive to reach—and which require the agents to do more work than they would have had to do in isolation. Mechanisms for dealing with state oriented domains are examined in [76-78,83,84]. Again, negotiation mechanisms that have certain attributes in task oriented domains (e.g., efficiency, stability) do not necessarily have these same attributes in state oriented domains.

6.3. Worth oriented domains

Worth oriented domains (WODs) are a generalization of state oriented domains, where agents assign a worth to each potential state, which establishes its desirability for the agent (as opposed to an SOD, in which the worth function is essentially binary—all non-goal states have zero worth). This establishes a decision theoretic flavor to interactions in a WOD. One example of a WOD is the TileWorld, as presented in [52]. The key advantage of a worth oriented domain is that the worth function allows agents to compromise on their goals, sometimes increasing the overall efficiency of the agreement. Every SOD can be cast in terms of a WOD, of course (with binary worth function). Negotiation mechanisms suitable for an SOD need not be suitable for a WOD (that is, the attributes of the same mechanism may change when moving from an SOD to a WOD). Mechanisms for dealing with worth oriented domains are examined in [79, 85].

7. The building blocks of a negotiation mechanism

Designing a negotiation mechanism, the overall "rules of interaction", is a three-step process. First, the agent designers must agree on a definition of the *domain*, then agree on a *negotiation protocol*, and finally propose a *negotiation strategy*.

7.1. Domain definition

The complete definition of a domain should give a precise specification to the concept of a goal, and to the agent operations that are available. For example, in the postmen domain, the goal of an agent is the set of letters that the agent must deliver (as in any TOD, the goal is the set of tasks that need to be carried out), along with the requirement that the agent begin and end at the post office.

The specification of agent operations that are available define exactly what an agent can do, and the nature of those actions' cost. In the postmen domain, again, it is part of the domain definition that an agent can carry an unlimited number of letters, and that the cost of a graph traversal is the total distance traveled.

This formal domain definition is the necessary first step in analyzing any new domain. If agents are negotiating over sharing message traffic in telecommunications networks, it is necessary to specify completely what constitutes a goal, and what agent operations are available. Similarly, PDAs involved in negotiations over schedules need their goals and operators precisely defined.

7.2. Negotiation protocol

Once the domain has been specified, we need to specify the negotiation protocol, which establishes the rules of interaction among agents. Here, we need to be concerned both with the *space of possible deals*, and with the *negotiation process*.

- *Space of possible deals*: First, we must specify the set of candidate deals. Specifically, what kinds of agreements can the agents come to? For example, we might restrict our agents to only discussing deals that do not involve redundant work (e.g., in the carpool example, the parents will not consider deals that have two parents visiting the same school). Similarly, we might specify that deals cannot involve tossing a coin.
- *Negotiation process*: Given a set of possible deals, what is the process that agents can use to converge to agreement on a single deal? In other words, what are the rules that specify how consensus will be reached? How will one agreed-upon deal be differentiated from the other candidates? In the carpool example, we might specify that each parent will in turn offer a delivery schedule and assignments; the next parent can either accept the offer, or reject it and make his own counter-offer. We might also allow as part of the negotiation process that any parent can, at any point, make a "take-it-or-leave-it" proposition, that will either be accepted or end the negotiation without agreement. Another example of a negotiation process was the monotonic concession protocol that was described in Section 2.5.1 above.

7.3. Negotiation strategy

Given a set of possible deals and a negotiation process, what strategy should an individual agent adopt while participating in the process? For example, one strategy for a parent in the carpool scenario is to compute a particular delivery schedule and present it as a "take-it-or-leave-it" deal. Another strategy is to start with the deal that is best for you, and if the other parent rejects it, minimally modify it as a concession to the other parent.

The specification of a negotiation strategy is not strictly part of the interaction rules being decided on by the designers of automated agents. In other words, the designers are really free to build their agents as they see fit. No one can compel them to build their agents in a certain way (having a certain strategy), and such compulsion, if attempted, would probably not be effective. However, we can provide strategies with known properties, and *allow* designers to incorporate them. More specifically, we may be able to bring to the table a given strategy, and show that it is provably optimal (for the agent itself). There will be no incentive for any designer to use any different strategy. And when all agents use that strategy, there will be certain (beneficial) global properties of the interaction. So a negotiation strategy is provided to the designers as a service; if a compelling case is made, the designers will in fact incorporate that strategy into their agents. We generally are interested in negotiation protocol/strategy combinations.

8. Three classes of TODs

As mentioned above, the domain examples given in Section 6.1 are all TODs, and seem to have a great deal in common with one another. There are, however, critical differences among them, all focused on the domains' cost functions. To demonstrate these differences, we categorize TODs based on three possible attributes of the cost function: *subadditivity*, *concavity*, and *modularity*. This is a hierarchy; modularity implies concavity, which in turn implies subadditivity. Protocols and strategies that are stable in one kind of TOD are not necessarily stable in other kinds. These issues are discussed at greater length below.

8.1. Subadditive

In some domains, by combining sets of tasks we may reduce (and can never increase) the total cost, as compared with the sum of the costs of achieving the sets separately. The postmen domain, for example, is subadditive. If X and Y are two sets of addresses, and we need to visit all of them ($X \cup Y$), then in the worst case we will be able to do the minimal cycle visiting the X addresses, then do the minimal cycle visiting the Y addresses. This might be our best plan if the addresses are disjoint and decoupled (the topology of the graph is against us). In that case, the cost of visiting all the addresses is equal to visiting one set plus the cost of visiting the other set. However, in some cases we may be able to do better, and visit some addresses on the way to others. That's what subadditivity means.

As another example, consider the database query domain. To evaluate two sets of queries, X and Y, we can of course evaluate all the queries in X, then independently evaluate all the queries in Y. This, again, might be our best course of action if the queries are disjoint and decoupled; the total cost will be the cost of X plus the cost of Y. However, sometimes we will be able to do better, by sharing the results of queries or subqueries, and evaluate $X \cup Y$ at lower total cost.

A relatively minor change in a domain definition, however, can eliminate subadditivity. If, in the postmen domain, the agents were not required to return to the post office at the end of their deliveries, then the domain would not be subadditive.

8.2. Concave

In a concave domain, the cost that arbitrary set of tasks Z adds to set of tasks Y cannot be greater than the cost Z would add to a subset of Y. The fax domain and the database query domain are concave, while the postmen domain is not. Intuitively, a concave domain is more "predictable" than a subadditive domain that is not concave. There is an element of monotonicity to the combining of tasks in a concave domain that is missing from non-concave domains. You know, for example, that if you have an original set of tasks (X), and are faced with getting an additional outside set (Z), you will not suffer greatly if you enlarge the original set—the extra work that Z adds will either be unaffected or reduced by the enlargement of the original set. In a non-concave domain, even if it is subadditive, you might find that the extra work that Z adds is much greater than it would have been before the enlargement.

8.3. Modular

In a modular domain, the cost of the combination of two sets of tasks is exactly the sum of their individual costs minus the cost of their intersection. This is, intuitively, the most well-behaved subadditive domain category of all. When task sets are combined, it is only their overlap that matters—all other tasks are extraneous to the negotiation. Only the fax domain from the above TOD examples is modular.

9. Incomplete information

Much of the research that we have been conducting on this model of negotiation considers issues relating to agents that have incomplete information about their encounter [78]. For example, they may be aware of their own goal without knowing the goal of the agent with whom they are negotiating. Thus, they may need to adapt their negotiation strategy to deal with this uncertainty.

One obvious way in which uncertainty can be exploited can be in misrepresenting an agent's true goal. In a task oriented domain, such misrepresentation might involve hiding tasks, or creating false tasks (phantoms, or decoys), all with the intent of improving one's negotiation position. The process of reaching an agreement generally depends on agents declaring their individual task sets, and then negotiating over the global set of declared tasks. By declaring one's task set falsely, one can in principle (under certain circumstances), change the negotiation outcome to one's benefit. Much of our research has been focused on negotiation mechanisms that disincentivize deceit. These kinds of negotiation mechanisms are called "incentive compatible" mechanisms in the game theory literature. When a mechanism is incentive compatible, no agent designer will have any reason to do anything but make his agent declare his true goal in a negotiation. Although the designer is free to build his agent any way he pleases, telling the truth will be shown to be the optimal strategy.

This concern for honesty among agents, and for encouraging that honesty by the very structure of the negotiation environment, is an absolutely essential aspect of work on multiagent systems. Situations in which agents have an incentive to lie are, in general, not stable. Although agent designers may discuss a strategy, they will then be motivated to go back and build their agents differently. This will ultimately result in less efficient systems (and outcomes that are worse for the individual agents). First, agents might reasonably expend a great deal of energy in discovering the true goal of the other negotiator, and all of this effort lowers the simplicity and efficiency of the system. Second, they will be tempted to risk strategies that may result in inferior outcomes. Two agents, coming together, each trying to outguess the other, will sometimes make choices that benefit no one.

Thus efficiency and stability are closely related. There is no point, in multiagent systems, in considering efficiency without considering stability. Without stability, efficiency cannot be guaranteed, as agents are tempted to deviate from the globally efficient strategy.

10. Task oriented domains—the formal definition

A *task oriented domain* (TOD) describes a certain class of scenarios for multiagent encounters. Intuitively, it is a domain that is cooperative, with no negative interactions among agents' goals. Each agent welcomes the existence of other agents, for they can only benefit from one another (if they can reach agreement about sharing tasks).

Definition 1. A *task oriented domain* (TOD) is a tuple $\langle \mathcal{T}, \mathcal{A}, c\rangle$ where:

(1) $\mathcal{T}$ is the set of all possible tasks.

(2) $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ is an ordered list of agents.

(3) c is a *monotonic* function $c : [2^{\mathcal{T}}] \to \mathbb{R}^+$. $[2^{\mathcal{T}}]$ stands for all the finite subsets of $\mathcal{T}$. For each finite set of tasks $X \subseteq \mathcal{T}$, $c(X)$ is the cost of executing all the tasks in X by a *single* agent. c is monotonic, i.e., for any two finite subsets $X \subseteq Y \subseteq \mathcal{T}$, $c(X) \leqslant c(Y)$.

(4) $c(\emptyset) = 0$.

Definition 2. An *encounter* within a TOD $\langle \mathcal{T}, \mathcal{A}, c\rangle$ is an ordered list $(T_1, T_2, \ldots, T_n)$ such that for all $k \in \{1, \ldots, n\}$, T_k is a *finite* set of tasks from $\mathcal{T}$ that A_k needs to achieve. T_k will also be called A_k's *goal*.

According to the definition above, the cost function c takes no parameters other than the task set. In general, c might be defined as having other, global, parameters (like the initial state of the world). However, the cost of a set of tasks is independent of others' tasks that need to be achieved. An agent in a TOD is certain to be able to achieve his goal at that cost.

11. Attributes and examples of TODs

Here we give several examples of TODs, which cover a variety of agent interaction situations. Subsequently, we will further classify each of these TOD examples with respect to the cost properties mentioned above in Section 8.

11.1. Delivery domain

Description: Agents have to deliver sets of containers to warehouses, which are arranged on a weighted graph $G = G(V, E)$. There is no limit to the number of containers that can fit in a warehouse. The agents all start from a central distribution point. Agents can exchange containers at no cost while they are at the distribution point, prior to delivery.

Task set: The set of all addresses in the graph, namely V. If address x is in an agent's task set, it means that he has at least one container to deliver to x.

Cost function: The cost of a subset of addresses $X \subseteq V$, i.e., $c(X)$, is the length of the minimal path that starts at the distribution point and visits all members of X.

11.2. Postmen domain

Description: Agents have to deliver sets of letters to mailboxes, which are arranged on a weighted graph $G = G(V, E)$. There is no limit to the number of letters that can fit in a mailbox. After delivering all letters, agents must return to the starting point (the post office). Agents can exchange letters at no cost while they are at the post office, prior to delivery.

Task set: The set of all addresses in the graph, namely V. If address x is in an agent's task set, it means that he has at least one letter to deliver to x.

Cost function: The cost of a subset of addresses $X \subseteq V$, i.e., $c(X)$, is the length of the minimal path that starts at the post office, visits all members of X, and ends at the post office.

11.3. Database queries

Description: Agents have access to a common database, and each has to carry out a set of queries. The result of each query is a set of records. For example, agent A_1 may want the records satisfying the condition "All female employees of company X earning over \$50,000 a year", and agent A_2 may want the records satisfying the condition "All female employees of company X with more than 10 years of seniority". Agents can exchange results of queries and subqueries at no cost.

Task set: All possible queries, expressed in the primitives of relational database theory, including operators like join, projection, union, intersection, and difference.

Cost function: The cost of a set of queries is the minimal number of database operations needed to generate all the records. It is possible to use the result of one query as input to other queries, i.e., the operations are not destructive.

11.4. The fax domain

Description: Agents are sending faxes to locations on a telephone network (a weighted graph). To send a fax, an agent must establish a connection with the receiving node. Once the connection is established, multiple faxes can be sent at no extra cost. The agents can, at no cost, exchange messages to be faxed.

Task set: The set of all possible receiving nodes in the network. If node x is in an agent's task set, it means that he has at least one fax to send to x.

Cost function: There is a cost associated with establishing a single connection to any node x. The cost of a set of tasks is the sum of the costs of establishing connections to all the nodes in the set. Thus, the cost of a dial-up connection to a given node is independent of other nodes in the task set.

11.5. Black/white hole domain

Description: There are a set of identical pegs, a basket, and a set of holes. A peg can be in a hole, in the basket, or in an agent's hand. Each hole is labeled as either black or white, and can hold at most one peg. One agent is only concerned with the configuration of pegs in black holes, and the other is only concerned with the configuration of pegs in white holes. Each agent has a list of (his own color) holes to fill or empty. There are enough pegs to satisfy both agents' goals. There are two operations in this world: PickUp (pick up the peg from a non-empty hole or any peg from the basket), and PutDown (put down the peg which is currently being held into an empty hole or the basket). An agent can hold no more than one peg at a time.

Task set: A single task (s, h) is to have hole s contain h pegs, where h is 0 or 1 (i.e., make hole s empty or have it contain one peg only). $\mathcal{T} = (s, h)$: s is a hole, $h \in \{0, 1\}$. In each encounter the world is in some initial state (this is an encounter-specific parameter that affects the cost function, but the cost function is still identical for all agents). One agent has a consistent task set that includes only white holes, while the second agent has a consistent task set that includes only black holes.

Cost function: The cost of a consistent set of tasks X is the minimal set of PickUp, PutDown operations that need to be done to move the world from its initial state to a final state that achieves all tasks in X. There is at least one minimal cost plan that does not involve the other agent's holes (neither takes nor places pegs into those holes).

11.6. Subclassification

Having introduced the TODs above, we now turn our attention to attributes that these domains exhibit. These attributes strongly affect their relationships to negotiation mechanisms. We will focus on the attributes of *subadditivity*, *concavity*, and *modularity*. The motivation for these definitions are presented in more detail below.

Definition 3 (*Subadditivity*). TOD $\langle \mathcal{T}, \mathcal{A}, c\rangle$ will be called *subadditive* if for all finite sets of tasks $X, Y \subseteq \mathcal{T}$, we have $c(X \cup Y) \leqslant c(X) + c(Y)$.

In other words, by combining sets of tasks we may reduce (and can never increase) the total cost, as compared with the cost of achieving the sets alone. All the TOD examples above are subadditive, except the delivery domain. To see why the delivery domain is not subadditive, consider the encounter shown in Fig. 1.

The cost of task a (i.e., the cost of delivering something to node a from the distribution point) is 1, which is also the cost of task b. However, the cost of the union of a and b is 3 (the deliverer, after delivering to one, must go back to the distribution point before continuing to deliver to the other). Thus the cost of the union of the tasks (3) is greater than the sum of the costs of the individual tasks (2). In the delivery domain there can be non-subadditive encounters.

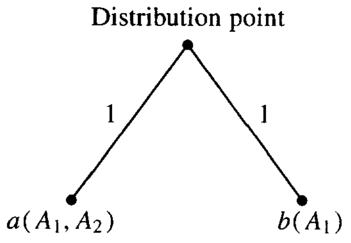

Fig. 1. Non-subadditive encounter in the delivery domain.

In this article, we are mainly concerned with two-agent subadditive domains.

Definition 4 (*Concavity*). TOD $\langle \mathcal{T}, \mathcal{A}, c\rangle$ will be called *concave* if for all finite sets of tasks $X \subseteq Y$, $Z \subseteq \mathcal{T}$, we have $c(Y \cup Z) - c(Y) \leqslant c(X \cup Z) - c(X)$.

In other words, the cost that an arbitrary set of tasks Z adds to a set of tasks Y cannot be greater than the cost Z would add to a subset of Y.

Theorem 5. *All concave TODs are also subadditive.*

The proof of this theorem and all other theorems, as well as additional lemmas, can be found in Appendix A.

The general postmen domain, and the black/white hole domain, are *not* concave. The other TOD examples (the fax domain and the database query domain) are concave.

The postmen domain is not concave because there may be cases such that a task (or a set of tasks) can add more to some other set of tasks than it adds to a subset of it. Such an example can be seen in Fig. 2. Every edge between nodes has cost 1.

Let $X = \{d, e, f, g\} \subset Y = \{b, c, d, e, f, g\}$ and $Z = \{h\}$.

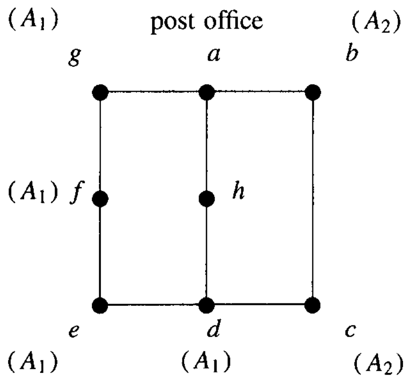

Fig. 2. Non-concave encounter in the postmen domain.

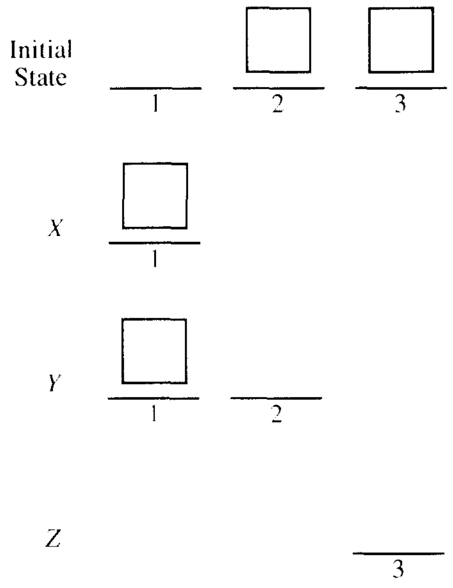

Fig. 3. Non-concave encounter in the black/white hole domain.

$$c(X\cup Z)=c(\{d,e,f,g,h\})=c(\{d,e,f,g\})=c(X)=6,$$

$$c(Y\cup Z)=c(\{b,c,d,e,f,g,h\})=9>c(\{b,c,d,e,f,g\})=c(Y)=7.$$

Therefore, we have a violation of the concavity condition:

$$2=c(Y\cup Z)-c(Y)>c(X\cup Z)-c(X)=0.$$

An example of the non-concavity of the black/white hole domain can be seen in Fig. 3.

In the initial state (as can be seen in the top of Fig. 3) slot 1 is empty while slots 2 and 3 are filled. X includes the task $(1,1)$ (i.e., slot 1 is filled). The cost of achieving X from the initial state ($c(X)$) is 2. All we need to do is to PickUp one block from the basket or from other slots and then to PutDown the block in slot 1. Y includes X and an additional task to empty slot 2 (i.e., $Y=\{(1,1),(2,0)\}$). To achieve Y we also need to do only one PickUp (this time we will pick the block in slot 2 up) and one PutDown (i.e. $c(Y)=2$). Z includes one task: to empty slot 3 (i.e., $Z=\{(3,0)\}$). For this reason, $c(X\cup Z)=2$, which is achieved by picking up the block in slot 3 and putting it down in the empty slot 1. However, $c(Y\cup Z)=4$, because we will need to move one block (from slot 2 to slot 1) to achieve Y, and then we will need to move another block (from slot 3 to the basket) to achieve Z. Again, we have a violation of the concavity condition:

$$2=c(Y\cup Z)-c(Y)>c(X\cup Z)-c(X)=0.$$

Theorem 6. *The postmen domain, restricted to graphs that have a tree topology (no cycles), is concave.*

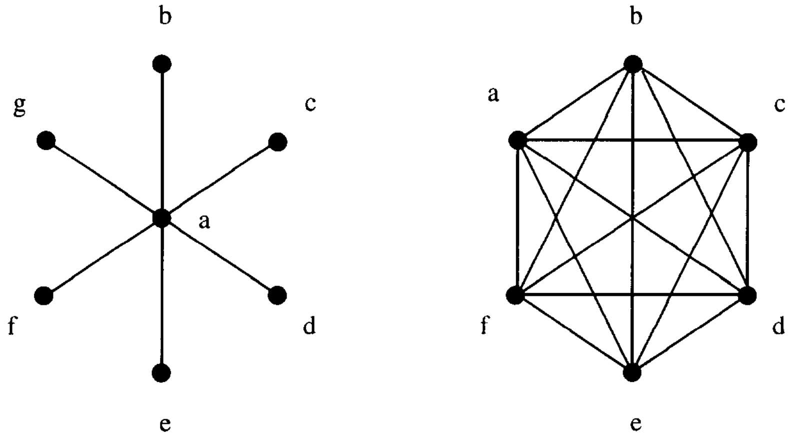

Fig. 4. Modular graph topologies.

Definition 7 (*Modularity*). TOD $\langle \mathcal{T}, \mathcal{A}, c\rangle$ will be called *modular* if for all finite sets of tasks $X, Y \subseteq \mathcal{T}$, we have $c(X \cup Y) = c(X) + c(Y) - c(X \cap Y)$.

In other words, the cost of the combination of two sets of tasks is exactly the sum of their individual costs minus the cost of their intersection.

Theorem 8. *All modular TODs are also concave.*

Only the fax domain from the above TOD examples is modular.

As stated in Theorem 6, the postmen domain restricted to a tree topology is concave. A graph has a *star* topology if there is no more than one node with degree greater than one (a star topology is a subcase of a tree topology). The node with degree greater than one is called the center of the graph. If we further restrict the postmen domain to trees that have star topologies (where the post office is at the center), then we have a modular TOD. A star example can be seen at the left side of Fig. 4, where the post office is at node a.

In a star topology the cost of visiting two different nodes $v, w \in \mathcal{T}$ is equal to the sum of visiting each node separately, i.e., $v \neq w \rightarrow c(\{v\} \cup \{w\}) = c(\{v\}) + c(\{w\})$. This implies the modularity condition.

The postmen domain restricted to fully connected (there is an arc between any two nodes) and homogeneous (all arcs have the same length) graphs, is also modular. An example of a fully connected and homogeneous graph can be seen in the right side of Fig. 4. If the length of all the nodes is 1 then the cost of visiting a set of nodes X in a fully connected and homogeneous graph is: $c(X) = |X| + 1$. This simply implies the modularity condition: $c(X \cup Y) = |X \cup Y| + 1 = |X| + |Y| - |X \cap Y| + 1 = (|X| + 1) + (|Y| + 1) - (|X \cap Y| + 1) = c(X) + c(Y) - c(X \cap Y)$.

Non-homogeneous fully connected graphs are not even concave. An example of that is the example seen above in Fig. 3, with additional sufficiently long additional arcs to make the graph fully connected.

12. Mechanisms for subadditive TODs

In this section, we develop the framework for formalizing two-agent negotiation mechanisms in subadditive task oriented domains. Similar definitions can be found in our previous work [75,77,78].

Definition 9. Given an encounter (T_1, T_2) within a two-agent TOD $\langle \mathcal{T}, \{A_1, A_2\}, c\rangle$ we have the following:

(1) A *pure deal* is a redistribution of tasks among agents. It is an ordered list (D_1, D_2) such that $D_1, D_2 \subseteq \mathcal{T}$, and $D_1 \cup D_2 = T_1 \cup T_2$. The semantics of such a deal is that each agent A_k commits itself to executing all tasks in D_k. The cost of such a deal to A_k is defined to be $\mathrm{Cost}_k(D_1, D_2) = c(D_k)$.

(2) A *mixed deal* is a pure deal (D_1, D_2) and a probability p, $0 \leqslant p \leqslant 1$. A mixed deal will be denoted by $(D_1, D_2)\colon p$. The semantics of this deal is that the agents will perform a lottery such that, with probability p, D_1 will be assigned to A_1 and D_2 will be assigned to A_2. With probability $1-p$, D_1 will be assigned to A_2 while D_2 will be assigned to A_1. The cost of such a deal to A_k is defined to be $\mathrm{Cost}_k((D_1, D_2)\colon p) = (p)c(D_k) + (1-p)c(D_{3-k})$.

(3) An *all-or-nothing* deal is a mixed deal $(T_1 \cup T_2, \emptyset)\colon p$. Agreeing to such a deal, A_1 has a p chance of executing all the tasks $T_1 \cup T_2$ and has a $1-p$ chance of doing nothing.

With the above definitions of three deal types, we now consider utility, the negotiation set, optimal protocols, and stable negotiation strategies.

Definition 10. Given an encounter (T_1, T_2) within a TOD $\langle \mathcal{T}, \{A_1, A_2\}, c\rangle$, we have the following:

(1) For any deal δ (pure, all-or-nothing, or mixed) we will define $\mathrm{Utility}_k(\delta) \equiv c(T_k) - \mathrm{Cost}_k(\delta)$.

(2) The (pure) deal $\Theta \equiv (T_1, T_2)$ will be called the *conflict deal*.

Θ is a conflict because no agent agrees to execute tasks other than its own. Note that for all k, $\mathrm{Utility}_k(\Theta) = 0$. When the agents fail to agree, i.e., run into a conflict, they by default execute the conflict deal Θ. Our assumption is that rational agents are utility maximizers; since they can guarantee themselves utility 0, they will not agree to any deal that gives them negative utility.

Definition 11. For vectors $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $\beta = (\beta_1, \beta_2, \ldots, \beta_n)$, we will say that α *dominates* β and write $\alpha \succ \beta$ if and only if $\forall k(\alpha_k \geqslant \beta_k)$, and $\exists l(\alpha_l > \beta_l)$. We will say that α *weakly dominates* β and write $\alpha \succeq \beta$ if and only if $\forall k(\alpha_k \geqslant \beta_k)$.

Definition 12. For deals δ and δ' (pure, all-or-nothing, or mixed), we will say that δ *dominates* δ', and write $\delta \succ \delta'$, if and only if $(\mathrm{Utility}_1(\delta), \mathrm{Utility}_2(\delta)) \succ (\mathrm{Utility}_1(\delta'), \mathrm{Utility}_2(\delta'))$. We will say that δ *weakly dominates* δ', and write $\delta \succeq \delta'$, if and only if $(\mathrm{Utility}_1(\delta), \mathrm{Utility}_2(\delta)) \succeq (\mathrm{Utility}_1(\delta'), \mathrm{Utility}_2(\delta'))$. We will say that δ is *equivalent* to δ', and write $\delta \equiv \delta'$ if $\forall k(\mathrm{Utility}_k(\delta) = \mathrm{Utility}_k(\delta'))$.

If $\delta \succ \delta'$ it means that the deal δ is better for at least one agent and not worse for the other.

Definition 13. Deal δ is *individual rational* if $\delta \succeq \Theta$.

A simple observation from the above definition and from Definition 10 (of the conflict deal and utility) is that a deal δ is individual rational if and only if $\forall k \in \{1,2\}$: $\text{Utility}_k(\delta) \geqslant 0$.

Definition 14. A deal δ is called *pareto optimal* if there does not exist another deal δ' such that $\delta' \succ \delta$ [33,43,60].

A pareto optimal deal cannot be improved upon for one agent without lowering the other agent's utility from the deal.

Definition 15. The set of all deals that are *individual rational* and *pareto optimal* is called the *negotiation set* (NS) [33].

Since agents are by definition indifferent between two deals that give them the same utility, we are interested in negotiation mechanisms that produce pareto optimal deals (i.e., if agent A_1 gets the same utility from deals x and y, but A_2 prefers y, we don't want them to settle on x). At this point, we are only considering negotiation mechanisms that result in a deal from the NS. These are, in some sense, mechanisms with efficient outcomes.

Theorem 16. *For any encounter in a TOD,* NS *over pure deals is not empty.*

Theorem 17. *For any encounter within any TOD,* NS *over mixed deals is not empty.*

13. Mechanisms that maximize the product of utilities

Having introduced the two mechanisms above (remember that a mechanism includes both a protocol and a strategy), we will shift our attention to the entire *class* of mechanisms that satisfy the following conditions:

- The protocol is symmetrically distributed.
- The strategy is in equilibrium with itself.
- Given the protocol, if two agents play the strategy they will agree on a deal that maximizes the product of their utilities. If there is more than one product maximizing deal, they will agree on a deal (among those product maximizers) that maximizes the sum of utilities. If there is more than one such sum maximizing product maximizer, the protocol will choose among those deals with some arbitrary probability. This definition implies both individual rationality and pareto optimality of the agreed-upon deals.

We will call this class of mechanisms the *product maximizing mechanisms*, or PMMs. All PMMs are efficient. Throughout the rest of this article, we will not be concerned with exactly which negotiation mechanism agents use, as long as it belongs to the PMM class.

Their are a number of existing approaches to the bargaining problem in game theory. One of the earliest and most popular was Nash's axiomatic approach [43, 50]. Nash was trying to axiomatically define a "fair" solution to a bargaining situation. He listed the following criteria as ones that a fair solution would satisfy:

(1) Individual rationality (it would not be fair for a participant to get less than he would anyway without an agreement).
(2) Pareto optimality (a fair solution will not specify an agreement that could be improved for one participant without harming the other).
(3) Symmetry (if the situation is symmetric, that is, both agents would get the same utility without an agreement, and for every possible deal, the symmetric deal is also possible, then a fair solution should also be symmetric, that is, give both participants the same utility).
(4) Invariance with respect to linear utility transformations. For example, imagine two agents negotiating on how to divide \$100. If one agent measures his utility in dollars while the other measures his in cents, it should not influence the fair solution. Similarly, if one agent already has \$10 in the bank, and evaluates the deal that gives him x dollars as having utility $10 + x$ while the other evaluates such a deal as having utility x, it should not influence the fair solution (i.e., change of origin doesn't affect the solution).
(5) Independence of irrelevant alternatives. Imagine two agents negotiating about how to divide 10,000 cents. The Nash solution will be 5,000 cents for each, due to the symmetry assumption above. Now imagine that the same agents are negotiating over \$100. Even though there are now some deals that they can't reach (for example, the one where one agent gets \$49.99, and the other gets \$50.01), the solution should be the same, because the original solution of 5,000 cents can still be found in the new deal space.

Nash showed that the product maximizing solution not only satisfies the above criteria, but is the only solution that satisfies them. The first four criteria above are explicitly or implicitly assumed in our own approach (in fact, for example, our version of the fourth assumption above is more restrictive than Nash's). The fifth criteria above is not *assumed* in our work, but turns out to be true in some cases anyway. We use the Nash solution, in general, as a reasonable bargaining outcome, when it is applicable. Nash, however, had some assumptions about the space of deals that we do not hold. For example, the Nash bargaining problem assumes a bounded, convex, continuous, and closed region of negotiation. In our agent negotiations, we do not assume that the space of deals is convex, nor that it is continuous.

Definition 18. A product maximizing mechanism (PMM) over a set of deals is a mechanism that has a negotiation strategy that is in equilibrium with itself—if all agents use this negotiation strategy, they will agree on a deal in NS that maximizes the product of the agents' utility [50]. If there is more than one such deal that maximizes the product, the mechanism chooses one arbitrarily, with equal probability.

Note that a PMM by definition satisfies the stability and efficiency criteria mentioned in Section 5.

The monotonic concession protocol defined above in Section 2.5.1 has an equilibrium strategy for each deal type that yields agreement on a deal in NS that maximizes the product of the agents' utility. Those strategies are based on Zeuthen risk criteria [74], and were presented in [75]. Therefore, the monotonic concession protocol (mentioned in Section 2.5.1) is a PMM.

Theorem 19. *A PMM over mixed deals in subadditive two-agent TODs divides the available utility equally between the two agents.*

14. Incentive compatible mechanisms

Sometimes agents do not have full information about one another's goals. This raises the question of whether agents can benefit from concealing goals, or manufacturing artificial goals. This lying can either occur explicitly, by declaring false goals, or implicitly, by behaving as if these false goals were true, depending on the specific negotiation mechanism. Our work in previous papers [75,77,78] partly focused on combinations of negotiation mechanisms and domains where agents have no incentive to lie. A negotiation mechanism is called *incentive compatible* when the strategy of telling the truth (or behaving according to your true goals) is in equilibrium (i.e., when one agent uses the strategy, the best thing the other agent can do is use the same strategy). In the postmen domain [75], we identified three types of lies:

(1) hiding tasks (e.g., a letter is hidden);
(2) phantom tasks (e.g., the agent claims to have a letter, which is non-existent and cannot be produced by the lying agent);
(3) decoy tasks (e.g., the agent claims to have a letter, which is non-existent but can be manufactured on demand if necessary).

Since certain deals might require the exchange of letters, a phantom lie can be uncovered, while a decoy lie (and of course a hidden lie) cannot. Thus, a phantom lie under certain negotiation mechanisms is "not safe". Different domains differ as to how easy or hard it is to generate decoy tasks.

In this section, we provide a characterization of the relationship between kinds of lies, domain attributes, and deal types. There are three kinds of lies in TODs, and we have considered three domain attributes (subadditivity, concavity, modularity) and three classes of optimal negotiation mechanisms, based on pure, all-or-nothing, and mixed deals. The resulting three-by-three-by-three matrix is represented in Fig. 5. Its notation is described below.

Consider the entry under subadditive, all-or-nothing deal, decoy lie (we'll refer to this as entry [a,j,z]). The entry L at that position means that for every optimal negotiation mechanism that is based on all-or-nothing deals, there exists a subadditive domain and

	Subadditive (a)			Concave (b)			Modular (c)		
	(x) Hid.	(y) Phan.	(z) Dec.	(x) Hid.	(y) Phan.	(z) Dec.	(x) Hid.	(y) Phan.	(z) Dec.
Pure (i)	L$\swarrow$	L$\swarrow$	L$\swarrow$	$\swarrow$L$\swarrow$	$\swarrow L_5^{\Rightarrow}$	$\swarrow$L	$\swarrow L_7$	T	$\Leftarrow T_4$
A-or-n (j)	$T_1^{\nearrow}$	T/P	$L_{6\Downarrow}$	$\nearrow$T$\nearrow$	T$\nearrow$	T$\nearrow$	$\nearrow$T	$\nearrow$T	$\nearrow$T
Mixed (k)	L$\swarrow$	$T/P_2^{\Uparrow}$	L	$\swarrow$L$\swarrow$	$T^{\Uparrow}\nearrow$	$\Leftarrow T_3^{\Uparrow}\nearrow$	$\swarrow L_8$	$\nearrow$T	$\nearrow$T

Fig. 5. Three-dimensional table of incentive compatibility.

an encounter such that at least one agent has the incentive to lie with a decoy lie (L means lying may be beneficial). The entry T at position [b, k, z] means that for every concave domain and every encounter within this domain, under any optimal negotiation mechanism based on mixed deals, agents do not have an incentive to lie with decoy lies (T means telling the truth is always beneficial).

The entries in the table marked T/P (such as [a, j, y]) refer to lies which are not beneficial because they may always be discovered (in any encounter within the domain); if the agent tells the truth, it is because he is afraid of the penalty that will be levied if his lie is discovered. Thus, T/P can be transformed into T if the optimal negotiation mechanism includes a sufficiently high penalty for discovered lies.

In the table, there is a relationship between cells. The fact that entry [a, j, x] is T implies that entry [b, j, x] will also be T (this is denoted by the $\nearrow$ single shaft arrow in the first cell, and the $\nearrow$ arrow going into the T in cell [b, j, x]). Similarly, [b, j, x] being T implies that entry [c, j, x] will be T (and is also denoted by arrows). This is because modular domains are concave, and concave domains are subadditive; if there is no incentive to lie even in a subadditive domain, there will certainly be no incentive to lie in concave or modular domains (which are subclasses of subadditive domains).

Similarly, the L entry in [c, i, x] implies that [b, i, x] will also have an L entry: if a beneficial lie can be found in a modular domain, then it can certainly be found in a concave domain (a superset). These downward influences of L are also marked in the table, with $\swarrow$ arrows.

There is also a relationship between certain table entries with the same domain attribute (these relationships are denoted by double shaft arrows like $\Leftarrow$). For example, if there is no incentive to lie in general mixed deals, there is no incentive to lie in all-or-nothing deals (which are a subset). Thus, the T in cell [b, k, z] implies the T in cell [b, j, z] (it also implies [b, k, y], which in turn implies [b, j, y], ...).

To fill out the table, therefore, we need only demonstrate a small number of "fixed points", which in turn imply all the other table entries. The fixed points that need to be demonstrated are numbered in the table from 1 to 8. We demonstrate the values for these 8 cells, and present some other theorems that make general statements about optimal negotiation mechanisms.

14.1. Incentive compatible fixed points

The four incentive compatible fixed points are determined by the theorems below.

Theorem 20 (Fixed point 1). *For any encounter in a two-agent subadditive TOD, and any optimal negotiation mechanism over all-or-nothing deals, every "hiding" lie is not beneficial.*

Theorem 21. *For any encounter in a two-agent subadditive TOD, there is always an all-or-nothing deal in* NS *maximizing the product of the utilities.*

Since an all-or-nothing deal is always a candidate agreement, the negotiation mechanism might arbitrarily choose it. This is the reason that the L in [a, k, z] is implied by the L in [a, j, z]. If there exists an encounter in which one of the agents has an incentive to lie when the negotiation is over all-or-nothing deals, then in the same encounter the same agents have the same incentive to lie when negotiating over general mixed deals: the agreement can always be an all-or-nothing deal.

Another consequence is the following theorem.

Theorem 22 (Fixed point 2). *For any encounter in a two-agent subadditive TOD, and any PMM over mixed deals, every "phantom" lie has a positive probability of being discovered. Therefore, with a sufficiently severe penalty mechanism, it is never beneficial to declare a phantom task.*

Theorem 23 (Fixed point 3). *For any encounter in a two-agent concave TOD, and any optimal negotiation mechanism over mixed deals, every "decoy" lie is not beneficial.*

Theorem 24. *For any encounter in a two-agent concave TOD, and any optimal negotiation mechanism over all-or-nothing deals, every lie (including combinations of hidden, phantom, and decoys) is not beneficial.*

Because of the theorem above, it is clear that for concave domains, agents cannot benefit by lying when all-or-nothing deals are in use—i.e., any optimal negotiation procedure over all-or-nothing deals is incentive compatible. This is also true for modular domains (a subcase). This can be seen in the table, where the entire all-or-nothing row is marked T for modular and concave domains.

Additionally, in a subadditive domain where decoy tasks cannot be generated, an optimal negotiation procedure over all-or-nothing deals with a penalty mechanism for discovered lies is incentive compatible (this was shown, in a different form, in [75]). This can be seen in the table, where the all-or-nothing row is marked with T and T/P (excluding the decoy column).

Theorem 25 (Fixed point 4). *For any encounter in a two-agent modular TOD, and any optimal negotiation mechanism over pure deals, every "decoy" lie is not beneficial.*

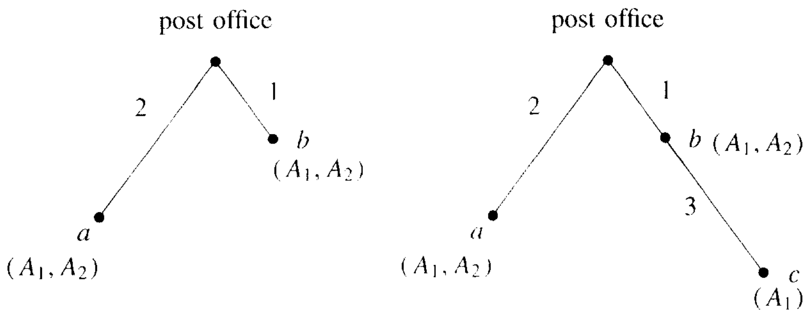

Fig. 6. Example of fixed point 5.

14.2. Non-incentive compatible fixed points

Fixed point 5

For an example of a beneficial phantom lie in a concave domain using a negotiation mechanism over pure deals, consider the following example in the postmen domain restricted to graphs that have a tree topology (this domain is concave, due to Theorem 6):

Example of a phantom letter. Consider the graph given on the left of Fig. 6 (the length, and thus the cost, of each edge is written next to it). The post office is at the root of the tree; both agents A_1 and A_2 need to deliver letters to nodes a and b.

Each agent has a 0.5 chance of delivering the letters to a (Utility = 2) and a 0.5 chance of delivering the letters to b (Utility = 4). The expected utility for both is 3.

What happens when A_1 creates a phantom letter, and tells A_2 that he has another letter to deliver to node c? See the right side of Fig. 6. The cost for A_1 of delivering his letters plus the phantom letter is now 12. It would not be individual rational for A_2 to visit c; A_1 will thus *have* to visit c, and he could deliver A_2's letter to b on his way. So they will agree on a deal where A_1 delivers the letters to b and c (with apparent utility of 4, and actual utility of 4). Thus, A_1's utility has risen from 3 to 4 by creating this phantom letter. This lie is also a "safe" lie, since A_2 cannot verify whether the phantom letter was actually delivered.

Fixed point 6

For an example of a beneficial decoy lie in a subadditive domain (e.g., the postmen domain with an unrestricted graph topology) using a negotiation mechanism over all-or-nothing deals, consider the following example.

Example. Let the graph be as in Fig. 7. Every edge between nodes has cost 1.

Agent A_1 needs to deliver letters to nodes d, e, f, and g, with a total cost of 6. Agent A_2 needs to deliver letters to nodes b and c, with a total cost of 4. If A_1 tells the truth, both utilities will be 1.5 (A_1 will do the whole cycle with probability $\frac{9}{14}$). After A_1 lies with producible decoy letter to node h, his apparent cost is still 6. Delivery of the entire set of letters, however, now (apparently) costs 9. They will agree on the deal that

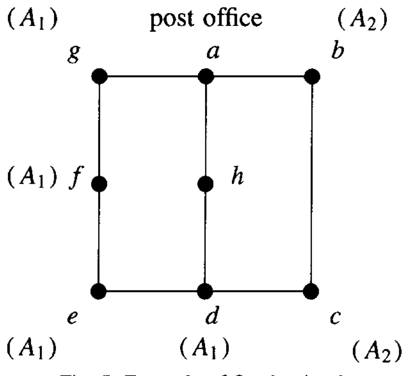

Fig. 7. Example of fixed point 6.

gives both agents apparent utility of 0.5. Agent A_2 will do the entire delivery, including the decoy letter, with probability $\frac{7}{18}$, in fact getting utility of 0.5. Agent A_1, however, will simply do the cycle (i.e., without the decoy letter), with probability $\frac{11}{18}$, getting an actual utility of 1.72. Lying is thus beneficial for A_1: telling the truth gives him a real utility of 1.5, while lying gives him a real utility of 1.72.

The example above is an instance of a more general case. It turns out that if an agent *knows* the relationship between his and his opponent's total costs in an all-or-nothing negotiation, it is possible for him to reliably generate a beneficial "default" lie that is made up of decoy tasks. In a subadditive domain, like the postmen domain, he can generate extra decoy tasks "along the way" (i.e., that don't increase his stand-alone cost), but which improve his position in the final agreement. This can only be done by A_1 when the cost of T_1 is greater than the cost of T_2. The example above is an instance of this situation. The important result here is that the lie can be generated reliably without having full information about the other agent's set of tasks.

Theorem 26. *For any encounter in a two-agent subadditive TOD, and any PMM over all-or-nothing deals, if agent A_1 knows that the cost of T_1 is greater than the cost of T_2, he can generate a default decoy lie that may benefit him, and will never harm him.*

To demonstrate fixed points 7 and 8, we bring two examples from the postmen domain that have as their topology that of a star; this is a modular TOD, and represented in Fig. 8. The post office is in the center of the star, and the length of the lines represent the distances from the post office.

Fixed point 7

Consider the left example in Fig. 8. Both agents have to deliver letters to nodes b (at a distance of 1) and e (at a distance of 2). Note that $c(T_1) = c(T_2) = c(T_1 \cup T_2) = 6$. If both agents tell the truth the negotiation mechanism will arbitrarily send one to node b and one to node e. If agent A_1 hides his letter to node b, then the only pure deal that maximizes the product of the agents' utilities is the one that sends agent A_1 to node b (!) and agent A_2 to node e. Thus, agent A_1 benefits from his lie.

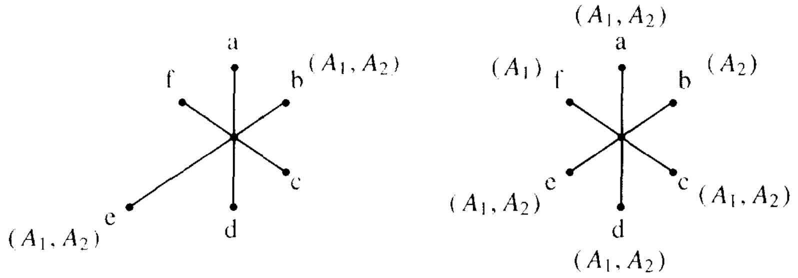

Fig. 8. Star topologies for postmen delivery.

Fixed point 8

Consider the right example in Fig. 8. Agent A_1 has to deliver a set of letters that includes ones to nodes a, c, d, e, and f. Agent A_2 has to deliver a set of letters that includes ones to nodes a, b, c, d, and e. Note that $c(T_1) = c(T_2) = 10$, and $c(T_1 \cup T_2) = 12$.

If A_1 hides his letter to node a, then let T_1^* be his apparent set of tasks (without node a). Note that $c(T_1^*) = 8$, and $c(T_1^* \cup T_2) = 12$. Using any PMM, under the assumption that A_1's true set of tasks is T_1^*, they can agree on a mixed deal $(X, Y){:}\,p$ such that $X = \{\text{a}, \text{f}\}$, $Y = \{\text{b}, \text{c}, \text{d}, \text{e}\}$, and $p = \frac{3}{4}$ (call this deal δ^*). Let U_1^* stand for the utility of A_1 from δ^*, and U_2^* stand for A_2's utility from that same deal. Utility $U_1^* = U_2^* = 3$. But A_1's real utility is $10 - \frac{3}{4}4 - \frac{1}{4}10 = 4.5$ which is greater than 4, the utility he would have gotten if he had told the truth.

The all-or-nothing deal is not beneficial for A_1 because the agents would agree on the probability $p = \frac{5}{12}$, which would give agent A_1 a real utility of $10 - \frac{5}{12}12 - \frac{5}{12}2 = 3\frac{5}{6} < 4$. However, the expected payoff for the lying agent is $4\frac{1}{6}$, i.e., still over 4, even when the negotiation mechanism sometimes chooses the all-or-nothing deal, so the lying agent benefits.

15. Conclusions

To provide agents with a suitable interaction environment in which they can coordinate, it may be desirable to establish high-level protocols that motivate socially beneficial (and individually beneficial) behavior. Game theory can provide tools appropriate to the design of these protocols. Some of the attributes that designers might like to see in interaction environments are efficiency, stability, and simplicity.

The design of suitable protocols is closely connected to the domain in which agents will be acting. Certain protocols might be appropriate for one domain, and inappropriate for another. In almost all cases, it is important to provide protocols that can deal with incomplete information on the part of agents, while maintaining the stability of the overall mechanism.

In this article we have presented a general domain theory to use in analyzing negotiation protocols. We have characterized task oriented domains (TODs), which cover an important set of multiagent interactions.

We have presented several examples of TODs, and examined three attributes that these domains can exhibit, namely subadditivity, concavity, and modularity. We have then enumerated the relationship between deal types, domain attributes, and types of deception, focusing on whether an agent in a TOD with a given attribute and deal type is motivated to always tell the truth. In particular, we have shown that in concave TODs, there is no benefit to an agent's lying when all-or-nothing deals are in use. In a general subadditive domain, however, when agents are able to generate decoy tasks, even all-or-nothing deals are not sufficient to create stability (discourage lies). In addition, we demonstrated that in subadditive domains, there often exist beneficial decoy lies that do not require full information regarding the other agent's goals.

Acknowledgements

Some of this research was done while Gilad Zlotkin was at the Computer Science Department of the Hebrew University of Jerusalem, and was supported by the Leibniz Center for Research in Computer Science. This research has also been partially supported by the Israeli Ministry of Science and Technology (Grant 032-8284) and by the Israel Science Foundation (Grant 032-7517).

Appendix A. Proofs

Theorem 5. *All concave TODs are also subadditive.*

Proof. The proof follows immediately if we choose X to be the empty set.

For any concave domain $\langle \mathcal{T}, \mathcal{A}, c\rangle$ and any finite subsets $Y, Z \subseteq \mathcal{T}$:

$$\begin{aligned} c(Y \cup Z) - c(Y) &\leqslant c(\emptyset \cup Z) - c(\emptyset) \\ &\leqslant c(Z), \\ c(Y \cup Z) &\leqslant c(Y) + c(Z). \qquad \square \end{aligned}$$

Theorem 6. *The postmen domain, restricted to graphs that have a tree topology (no cycles), is concave.*

Proof. Let $G(V, E)$ be a weighted, connected graph that has a tree topology. A graph has a tree topology if and only if it contain no cycles, i.e., there is at most one non-repetitive path (each node in the path appears only once) between any two nodes. Let $p \in V$ be the post office. For each subset of nodes $X \subseteq V$, $c(X)$ is the cost of the minimal cyclic path that starts at the post office, visits all the nodes in X, and ends at the post office. Since the graph has a tree topology, the minimal path that visits X is unique modulo the order of visiting specific nodes. Let $\hat{X}$ (called the closure of X) be the union of all nodes that are visited in the minimal path that visited X. $\hat{X}$ is also the maximal superset (in the $\subseteq$ relation) of X that has the same cost $c(X)$. $\hat{X}$ is also the minimal connected subgraph of G that includes both the post office p and X.

For a subset of nodes $X \subset V$ and a specific node $v \in V$, we define the distance between a set of nodes to a given node $dist(X, v)$ to be the length of the shortest path between v to one of the nodes in the closure of X ($\hat{X}$). If $a \in \hat{X}$ then $dist(X, v)$ is defined to be 0. Due to the fact that G has no loops, $c(X \cup \{v\}) = c(X) + 2dist(X, v)$.

To show that the domain is concave we need to show that for all sets of nodes $X \subseteq Y$, and $Y, Z \subseteq V$, we have $c(Y \cup Z) - c(Y) \leqslant c(X \cup Z) - c(X)$.

We will prove this by induction on the size of Z ($|Z|$).

Assume that $|Z| = 1$, i.e., $Z = \{z\}$.

We know that $c(X \cup \{z\}) = c(X) + 2dist(X, z)$ therefore: $c(X \cup \{z\}) - C(X) = 2dist(X, z)$. Using the same argument, we have that $c(Y \cup \{z\}) - C(Y) = 2dist(Y, z)$.

Since $X \subseteq Y$, we have that $\hat{X} \subseteq \hat{Y}$. This implies that $dist(Y, z) \leqslant dist(X, z)$, which is what we needed to show. The distance to a superset cannot be greater than the distance to a subset.

Assume that $c(Y \cup Z) - c(Y) \leqslant c(X \cup Z) - c(X)$ is true for sets Z of size less than or equal to n. We will then show that it is also true for sets of size $n + 1$.

Let $Z \subseteq V$ be an arbitrary set of nodes such that $|Z| = n$, and let $z \notin Z$ be some node in V. But then we have that:

(1) $c(Y \cup Z) - c(Y) \leqslant c(X \cup Z) - c(X)$ (from the induction asssumption);

(2) $c([Y \cup Z] \cup \{z\}) - c([Y \cup Z]) \leqslant c([X \cup Z] \cup \{z\}) - c([X \cup Z])$ (from the $n = 1$ case above, since $X \cup Z$ is a subset of $Y \cup Z$).

If we add both inequalities, we get:

$$c([Y \cup Z] \cup \{z\}) - c(Y) \leqslant c([X \cup Z] \cup \{z\}) - c(X),$$

or, regrouping, we have

$$c(Y \cup [Z \cup \{z\}]) - c(Y) \leqslant c(X \cup [Z \cup \{z\}]) - c(X).$$

Thus the set $Z \cup \{z\}$ also satisfies the concavity condition. □

Theorem 8. *All modular TODs are also concave.*

Proof. For any modular domain $\langle \mathcal{T}, \mathcal{A}, c \rangle$ and any finite subsets $X \subseteq Y$, $Z \subseteq \mathcal{T}$:

$$X \subseteq Y,$$

$$X \cap Z \subseteq Y \cap Z.$$

Monotonicity:

$$c(X \cap Z) \leqslant c(Y \cap Z),$$

Modularity:

$$c(X \cap Z) = c(X) + c(Z) - c(X \cup Z) \leqslant c(Y) + c(Z) - c(Y \cup Z) = c(Y \cap Z),$$

$$c(X) - c(X \cup Z) \leqslant c(Y) - c(Y \cup Z),$$

$$c(Y \cup Z) - c(Y) \leqslant c(X \cup Z) - c(X). \quad □$$

Theorem 16. *For any encounter in a TOD,* NS *over pure deals is not empty.*

Proof. Θ is an individual rational pure deal, and there cannot be an infinite chain of pure deals such that $\cdots \succ \delta'' \succ \delta' \succ \delta \succ \Theta$. This is because for any deal δ, and for any $k \in \{1,2\}$, $\text{Utility}_k(\delta)$ is bounded by $c(T_k)$ (since maximal benefit is reached when all of one's tasks are executed by the other agent). There exists at least one maximal pure deal δ^* that will be pareto optimal, but it is also individual rational, so $\delta^* \in \text{NS}$. □

Theorem 17. *For any encounter within any TOD,* NS *over mixed deals is not empty.*

Proof. Let $\langle T_1, T_2\rangle$ be some encounter in a two-agent TOD. To show that the negotiation set is not empty, it is sufficient to show a mixed deal that is both individual rational and pareto optimal.

Among all pure deals (D_1, D_2) that satisfy the following two conditions:

$$\textit{min:}\quad \min_{i=1}^{2} c(T_i) \geqslant \min_{i=1}^{2} c(D_i),$$

$$\textit{sum:}\quad \sum_{i=1}^{2} c(T_i) \geqslant \sum_{i=1}^{2} c(D_i),$$

let (D_1^*, D_2^*) be one that also has *minimal total* cost, i.e.,

$$c(D_1^*) + c(D_2^*) = \min_{(D_1,D_2)\text{ that satisfies the } \textit{min} \text{ and } \textit{sum} \text{ conditions}} c(D_1) + c(D_2).$$

Since the conflict deal $\Theta \equiv (T_1, T_2)$ satisfies the *min* and *sum* conditions and there are only a finite number of pure deals, the pure deal (D_1^*, D_2^*) exists.

Without loss of generality, we can assume that $c(T_2) \geqslant c(T_1)$ and $c(D_2^*) \geqslant c(D_1^*)$. From the *min* condition, we see that $c(T_1) \geqslant c(D_1^*)$. There are two cases:

- If $c(T_2) \geqslant c(D_2^*)$, then the deal $(D_1^*, D_2^*)\colon 1$ is individual rational.
- If $c(T_2) < c(D_2^*)$, then the deal $(D_1^*, D_2^*)\colon p$ (where $p = 1 - (c(T_1) - c(D_1^*))/(c(D_2^*) - c(D_1^*)))$ is individual rational.

$(D_1^*, D_2^*)\colon p$ is also pareto optimal, because if there is another deal $(D_1', D_2')\colon q$ that dominates $(D_1^*, D_2^*)\colon p$, then $(D_1', D_2')\colon q$ is also individual rational and therefore satisfies the *min* and *sum* conditions:

$$\text{Utility}_i((D_1^*, D_2^*)\colon p) \geqslant 0,$$

$$\begin{aligned} c(T_i) - (p)c(D_i^*) &\geqslant 0,\\ c(T_i) &\geqslant (p)c(D_i^*)\\ &\geqslant (p)c(D_i^*) + (1-p)c(D_{3-i}^*)\\ &\geqslant \min_{l\in\{1,2\}} c(D_l^*), \end{aligned}$$

$$\sum_{i\in\{1,2\}} c(T_i) \geqslant \sum_{i\in\{1,2\}} c(D_i^*),$$

$$\min_{i\in\{1,2\}} c(T_i) \geqslant \min_{i\in\{1,2\}} c(D_i^*).$$

Since $(D_1', D_2'):q$ dominates $(D_1^*, D_2^*):p$ it also implies that

$$\sum_{i\in\{1,2\}} \text{Utility}_i((D_1', D_2'):q) > \sum_{i\in\{1,2\}} \text{Utility}_i((D_1^*, D_2^*):p).$$

This can be true only if

$$\sum_{i\in\{1,2\}} c(D_i') < \sum_{i\in\{1,2\}} c(D_i^*).$$

But this contradicts the fact that (D_1^*, D_2^*) is the *minimal total cost* pure deal that satisfies the *min* and *sum* conditions. $\square$

Lemma A.1 (Efficiency of agreement). *For any encounter in a two-agent subadditive TOD, if a mixed deal $(D_1, D_2):p \in$ NS then $c(D_1) + c(D_2) = c(D_1 \cup D_2)$.*

Proof. $c(D_1) + c(D_2) \geqslant c(D_1 \cup D_2)$ because of the subadditivity of c. Using proof by contradiction, let's assume that $c(D_1) + c(D_2) > c(D_1 \cup D_2)$, then show that this results in a contradiction.

If $c(D_1) + c(D_2) > c(D_1 \cup D_2)$, then we can show that the deal $(D_1, D_2):p$ is dominated by the all-or-nothing deal $(D_1 \cup D_2, \emptyset):q$, such that

$$q = \frac{pc(D_1) + (1-p)c(D_2)}{c(D_1 \cup D_2)}.$$

This would contradict the fact that $(D_1, D_2):p$ is in the negotiation set, since it would then not be pareto optimal. Thus, if $c(D_1) + c(D_2) \geqslant c(D_1 \cup D_2)$ but cannot be greater, then the two sides of the equation must be equal.

First, we will show that q lies between 0 and 1, which it must in order for $(D_1 \cup D_2, \emptyset):q$ to be a legal mixed deal.

$$0 \leqslant c(D_1) \leqslant c(D_1 \cup D_2),$$

$$0 \leqslant c(D_2) \leqslant c(D_1 \cup D_2),$$

$$0 \leqslant pc(D_1) + (1-p)c(D_2) \leqslant c(D_1 \cup D_2).$$

Thus $0 \leqslant q \leqslant 1$, by the definition of q.

The following completes the proof by demonstrating the above claim that the all-or-nothing deal $(D_1 \cup D_2, \emptyset):q$ dominates the deal $(D_1, D_2):p$.

$$\begin{aligned}\text{Cost}_1((D_1 \cup D_2, \emptyset):q) &= qc(D_1 \cup D_2) \quad \text{(by the definition of } q\text{)}\\ &= pc(D_1) + (1-p)c(D_2)\\ &= \text{Cost}_1((D_1, D_2):p).\end{aligned}$$

So for agent A_1, the two deals are equivalent. Because of the subadditivity condition, i.e., that $c(D_1) + c(D_2) > c(D_1 \cup D_2)$, it follows that $1 < (c(D_1) + c(D_2))/(c(D_1 \cup D_2))$. By splitting up the right side of the equation, we get

$$1 < \frac{pc(D_1) + (1-p)c(D_2)}{c(D_1 \cup D_2)} + \frac{(1-p)c(D_1) + pc(D_2)}{c(D_1 \cup D_2)}.$$

Since the first term is equal to q, this means that

$$1 < q + \frac{(1-p)c(D_1) + pc(D_2)}{c(D_1 \cup D_2)},$$

$$(1-q) < \frac{(1-p)c(D_1) + pc(D_2)}{c(D_1 \cup D_2)},$$

$$(1-q)c(D_1 \cup D_2) < (1-p)c(D_1) + pc(D_2).$$

And so, because the left side of the inequality equals $\text{Cost}_2((D_1 \cup D_2, \emptyset){:}\, q)$, and the right side of the inequality equals $\text{Cost}_2((D_1, D_2){:}\, p)$, we have the following: $\text{Cost}_2((D_1 \cup D_2, \emptyset){:}\, q) < \text{Cost}_2((D_1, D_2){:}\, p)$.

Thus the all-or-nothing deal is cheaper for agent A_2, and therefore agent A_2's utility is greater. We have

$$\text{Utility}_1((D_1 \cup D_2, \emptyset){:}\, q) = \text{Utility}_1((D_1, D_2){:}\, p),$$

$$\text{Utility}_2((D_1 \cup D_2, \emptyset){:}\, q) > \text{Utility}_2((D_1, D_2){:}\, p),$$

which means that $(D_1 \cup D_2, \emptyset){:}\, q \succ (D_1, D_2){:}\, p$. Thus the all-or-nothing deal dominates. □

Lemma A.2 (Constant sum equality). *For any encounter in a two-agent subadditive TOD, for each mixed deal $\delta \in$ NS, $\text{Utility}_1(\delta) + \text{Utility}_2(\delta) = c(T_1) + c(T_2) - c(T_1 \cup T_2)$.*

Proof. Let $\delta = (D_1, D_2){:}\, p$ be a mixed deal.

$$\begin{aligned}
&\text{Utility}_1(\delta) + \text{Utility}_2(\delta)\\
&\quad = c(T_1) - [pc(D_1) + (1-p)c(D_2)] + c(T_2) - [(1-p)c(D_1) + pc(D_2)]\\
&\qquad \text{(Definition of utility)}\\
&\quad = c(T_1) + c(T_2) - [c(D_1) + c(D_2)]\\
&\quad = c(T_1) + c(T_2) - c(D_1 \cup D_2) \quad \text{(Lemma A.1)}\\
&\quad = c(T_1) + c(T_2) - c(T_1 \cup T_2) \quad \text{(Definition of a deal).} \qquad \square
\end{aligned}$$

Theorem 19. *A PMM over mixed deals in subadditive two-agent TODs divides the available utility equally between the two agents.*

Proof. For any deal $\delta \in$ NS, $C \equiv \text{Utility}_1(\delta) + \text{Utility}_2(\delta) = c(T_1) + c(T_2) - c(T_1 \cup T_2)$ because of Lemma A.2. From the definition of a PMM we know that the agreement that will be reached will maximize the product of their utilities. Let δ be a deal that the agents agree on after negotiation when both are using a PMM.

$$\begin{aligned}\pi(\delta) &\equiv \text{Utility}_1(\delta)\text{Utility}_2(\delta)\\ &= \text{Utility}_1(\delta)(C - \text{Utility}_1(\delta))\\ &= \text{Utility}_1(\delta)C - \text{Utility}_1(\delta)^2.\end{aligned}$$

To maximize π, we take the derivative of π with respect to $\text{Utility}_1(\delta)$. To discover what value of $\text{Utility}_1(\delta)$ will maximize π, the first derivative should equal zero. $(\text{Utility}_1(\delta)C - \text{Utility}_1(\delta)^2)' = C - 2\text{Utility}_1(\delta) = 0$ means that $\text{Utility}_1(\delta) = \frac{1}{2}C = \text{Utility}_2(\delta)$. □

Theorem 20 (Fixed point 1). *For any encounter in a two-agent subadditive TOD, and any PMM over all-or-nothing deals, every "hiding" lie is not beneficial.*

Proof. Assume that agent A_1 decides to hide one of his tasks; we will show that this cannot be beneficial using all-or-nothing deals.

Let $a = c(T_1)$, $b = c(T_2)$ and $c = c(T_1 \cup T_2)$. Let $\delta(p)$ be the all-or-nothing deal $(T_1 \cup T_2, \emptyset) : p$. The agents will agree on this all-or-nothing deal $\delta(p)$ such that $\text{Utility}_1(\delta(p)) = \text{Utility}_2(\delta(p))$ (Theorem 19).

$$\text{Utility}_1(\delta(p)) = a - \text{Cost}_1(\delta(p)) = b - \text{Cost}_2(\delta(p)) = \text{Utility}_2(\delta(p)),$$

$$a - pc = b - (1-p)c,$$

$$p = \frac{c+a-b}{2c},$$

$$(1-p) = \frac{c+b-a}{2c}.$$

So the agreement reached is $\gamma \equiv \delta((c+a-b)/(2c))$.

$$\text{Cost}_1(\gamma) = \frac{c+a-b}{2},$$

$$\text{Cost}_2(\gamma) = \frac{c+b-a}{2},$$

$$\text{Utility}_1(\gamma) \text{ and } \text{Utility}_2(\gamma) = \frac{a+b-c}{2}.$$

If agent A_1 lies and hides some tasks $L \subseteq T_1$ and broadcasts $T_1^* = T_1 - L$, then (because of the subadditivity of the domain)

$$c(T_1) \leqslant c(T_1^*) + c(L), \tag{A.1}$$

$$c(T_1 \cup T_2) \leqslant c(T_1^* \cup T_2) + c(L). \tag{A.2}$$

Let $a^* = c(T_1^*)$, $l = c(L)$ and $c^* = c(T_1^* \cup T_2)$. If they negotiate on T_1^* and T_2 they will agree on $[(T_1^* \cup T_2, \emptyset) : (c^* + a^* - b)/(2c^*)]$. Agent A_1 will perform $T_1 \cup T_2$ with probability $(c^* + a^* - b)/(2c^*)$ or perform only L with probabilty $(c^* + b - a^*)/(2c^*)$ and his expected cost will be:

$$\frac{c^* + a^* - b}{2c^*}c + \frac{c^* + b - a^*}{2c^*}l.$$

To complete the proof, it will be sufficient to show that

$$\frac{c^* + a^* - b}{2c^*}c + \frac{c^* + b - a^*}{2c^*}l \geqslant \frac{c + a - b}{2},$$

$$c(c^* + a^* - b) + l(c^* + b - a^*) \geqslant c^*(c + a - b),$$

$$cc^* + c(a^* - b) + l(c^* + b - a^*) \geqslant c^*c + c^*(a - b),$$

$$c(a^* - b) + l(c^* + b - a^*) \geqslant c^*(a - b).$$

There are two cases:

Case 1: $c + a^* \geqslant c^* + a$.

Case 2: $c + a^* < c^* + a$.

Case 1. We know that $c^* + l \geqslant c$ (Eq. (A.2)) which means that $l \geqslant c - c^*$. It will be sufficient to show that:

$$c(a^* - b) + (c - c^*)(c^* + b - a^*) \geqslant c^*(a - b),$$

$$c(a^* - b + c^* + b - a^*) - c^*(c^* + b - a^*) \geqslant c^*(a - b),$$

$$cc^* - c^*(c^* + b - a^*) \geqslant c^*(a - b),$$

$$c - (c^* + b - a^*) \geqslant a - b,$$

$$c - c^* - b + a^* \geqslant a - b,$$

$$c - c^* + a^* \geqslant a,$$

$$c + a^* \geqslant a + c^*.$$

Case 2. $c + a^* < c^* + a$ means that $c^* > c + a^* - a$. It will be sufficient to show that:

$$c(a^* - b) + l(c + a^* - a + b - a^*) \geqslant c^*(a - b),$$

$$c(a^* - b) + l(c + b - a) \geqslant c^*(a - b),$$

$$c(a^* - b) + lc + l(b - a) \geqslant c^*(a - b),$$

$$c(a^* - b + l) + l(b - a) \geqslant c^*(a - b),$$

$$c(a^* - b + l) + (c^* + l)(b - a) \geqslant 0.$$

We know that $c^* + l \geqslant c$ (Eq. (A.2)). It will be sufficient to show that:

$$c(a^* - b + l) + c(b - a) \geqslant 0,$$

$$c(a^* - b + l + b - a) \geqslant 0,$$

$$c(a^* + l - a) \geqslant 0,$$

$$a^* + l - a \geqslant 0,$$

$$a^* + l \geqslant a,$$

which is true (Eq. (A.1)). □

Theorem 21. *For any encounter in a two-agent subadditive TOD, there is always an all-or-nothing deal in* NS *maximizing the product of the utilities.*

Proof. Assume that $(D_1, D_2):p$ is in the negotiation set. If we use the same probability q that was defined above (in the proof of Lemma A.1), we can show that $(D_1 \cup D_2, \emptyset):q \succeq (D_1, D_2):p$ using the same steps as in the proof of Lemma A.1, by changing the "$<$" symbol to "$\leqslant$". Because $(D_1, D_2):p$ is in the negotiation set, it cannot be dominated, and therefore $(D_1 \cup D_2, \emptyset):q \equiv (D_1, D_2):p$. □

Theorem 22 (Fixed point 2). *For any encounter in a two-agent subadditive TOD, and any PMM over mixed deals, every "phantom" lie has a positive probability of being discovered. Therefore, with a sufficiently severe penalty mechanism, it is never beneficial to declare a phantom task.*

Proof. This follows immediately from Theorem 21. The agents can always agree on some all-or-nothing deal, which will assign all tasks (including the phantom tasks) to the non-lying agent with some positive probability. □

Theorem 23 (Fixed point 3). *For any encounter in a two-agent concave TOD, and any PMM over mixed deals, every "decoy" lie is not beneficial.*

Proof. Let (T_1, T_2) be the true encounter. Let $a = c(T_1)$, $b = c(T_2)$, and $c = c(T_1 \cup T_2)$. Negotiation using the true information and using any PMM over mixed deals will lead to an agreement on a mixed deal δ that satisfies $\text{Utility}_1(\delta) = \frac{1}{2}(a+b-c) = \text{Utility}_2(\delta)$.

Assume that agent A_1 lies and declares that his initial set of tasks is $T_1^* = T_1 \cup L$, where L is a set of decoy tasks. Let $a^* = c(T_1^*)$ and $c^* = c(T_1^* \cup T_2)$; in other words, a^* is agent A_1's apparent initial cost, and c^* is his apparent total cost.

The negotiation that asssumes this information is true will lead to an agreement on a mixed deal δ^* that satisfies $\text{Utility}_1(\delta^*) = \frac{1}{2}(a^* + b - c^*) = \text{Utility}_2(\delta^*)$. Because $T_1^* \subseteq \{T_1^* \cup T_2\}$, we can use the definition of concavity and conclude that $c^* - c \leqslant a^* - a$ (because the cost that L adds to T_1 is greater than the cost that L adds to $\{T_1 \cup T_2\}$). This also means that $a - c \leqslant a^* - c^*$. We can thus conclude that $\text{Utility}_2(\delta^*) = \frac{1}{2}(a^* + b - c^*) \geqslant \frac{1}{2}(a + b - c) = \text{Utility}_2(\delta)$. In other words, even though agent A_2 might have to perform some or all of the decoy tasks, his utility can only be larger than if agent A_1 would have told the truth.

Because we are speaking of a constant sum game (Lemma A.2), it seems reasonable that if agent A_2's utility goes up, agent A_1's utility will go down.[3] This is what we will now show.

The agents will agree on a mixed deal $\delta^* = (X^*, Y^*): p$ such that $X^* \cup Y^* = T_1^* \cup T_2$. Let $X = X^* - L$, $Y = Y^* - L$, $x^* = c(X^*)$, $y^* = c(Y^*)$, $x = c(X)$ and $y = c(Y)$.

Agent A_1 does *not* have to perform his decoy tasks, however, and therefore his actual utility from δ^* (which we denote by $\overline{\text{Utility}_1}(\delta^*)$) is $a - px - (1-p)y$.

On the other hand, agent A_2 *does* have to perform the decoy tasks. Therefore his utility from δ^* is $b - (1-p)x^* - py^*$. We can sum those two utilities.

$$\begin{aligned}
\overline{\text{Utility}_1}(\delta^*) + \text{Utility}_2(\delta^*) &= a - px - (1-p)y + b - (1-p)x^* - py^* \\
&= a + b - ((1-p)y + py^*) + (px + (1-p)x^*) \\
&\leqslant a + b - (y + x) \\
&\leqslant a + b - c \\
&= \text{Utility}_1(\delta) + \text{Utility}_2(\delta).
\end{aligned}$$

Note that $X \subseteq X^*$, therefore $x \leqslant x^*$, and as a consequence, $x \leqslant px + (1-p)x^*$. Similarly, $Y \subseteq Y^*$, therefore $y \leqslant y^*$, and as a consequence, $y \leqslant (1-p)y + py^*$. Also, according to Theorem 5 (all concave TODs are also subadditive TODs) and the definition of subadditivity (since $Y \cup X = T_1 \cup T_2$), we have that $c \leqslant x + y$.

We have shown that agent A_1's utility plus agent A_2's utility when agent A_1 lies is less than or equal to their combined utility when agent A_1 doesn't lie. However, we have already shown that agent A_2's utility when agent A_1 lies is greater than or equal to agent A_2's utility when agent A_1 doesn't lie. Thus, the conclusion is that agent A_1's actual utility cannot be bigger than his unvarnished utility. The lie is not beneficial for agent A_1. □

Theorem 24. *For any encounter in a two-agent concave TOD, and any PMM over all-or-nothing deals, every lie (including combinations of hidden, phantom, and decoys) is not beneficial.*

Proof. Let (T_1, T_2) be the encounter, and let the agreement that will be reached using a PMM with the common knowledge of the true information be the mixed deal δ.

Assume that agent A_1 declares that his set of tasks is $T_1^* = \{T_1 - T_h \cup T_d\}$, where T_h is the set of all his hidden tasks, and T_d is the set of all his decoy tasks. Let the agreement that the agents will reach when they assume this information is true be δ^*.

Consider the negotiation where agent A_1 declares his set of tasks to be $T_1^{**} = \{T_1 - T_h\}$. Let δ^{**} be the agreement that the agents will reach in the above negotiation.

Let $a^{**} = c(T_1^{**})$ and $c^{**} = c(T_1^{**} \cup T_2)$. In other words, a^{**} is A_1's apparent cost, and c^{**} is the apparent total cost.

[3] Actually, it is more subtle: agent A_1's apparent utility also goes up along with A_2's, but what we are concerned with is A_1's actual utility.

The negotiation that asssumes this information is true will lead to an agreement on a mixed deal δ^{**} that satisfies: $\text{Utility}_1(\delta^{**}) = \frac{1}{2}(a^{**} + b - c^{**}) = \text{Utility}_2(\delta^{**})$. Because $T_1^{**} \subseteq \{T_1^{**} \cup T_2\}$, we can use the definition of concavity and conclude that $c - c^{**} \leqslant a - a^{**}$ (because the cost that L adds to T_1^{**} is greater than the cost that L adds to $\{T_1^{**} \cup T_2\}$). This also means that $a^{**} - c^{**} \leqslant a - c$, so we can conclude that $\text{Utility}_1(\delta^{**}) = \frac{1}{2}(a^{**} + b - c^{**}) \leqslant \frac{1}{2}(a + b - c) = \text{Utility}_1(\delta)$. In other words, the apparent utility of agent A_1 is less than his unvarnished utility without the hidden tasks.

Let us now consider the situation where agent A_1 declares $T_1^* = \{T_1^{**} \cup T_d\}$. From Theorem 23 we can see that $\overline{\text{Utility}_1}(\delta^*) \leqslant \text{Utility}_1(\delta^{**})$, and the final conclusion will be that $\overline{\text{Utility}_1}(\delta^*) \leqslant \text{Utility}_1(\delta)$. The liar's actual utility will be less than his unvarnished utility. □

Theorem 25 (Fixed point 4). *For any encounter in a two-agent modular TOD, and any PMM over pure deals, every "decoy" lie is not beneficial.*

Proof. Let (T_1, T_2) be an encounter. Let $\delta = (D_1, D_2)$ be a pure deal, i.e., $D_1 \cap D_2 = \emptyset$ and $D_1 \cup D_2 = T_1 \cup T_2$. $\text{Utility}_1(\delta) + \text{Utility}_2(\delta) = (c(T_1) - c(D_1)) + (c(T_2) - c(D_2)) = c(T_1) + c(T_2) - (c(D_1) + c(D_2))$.

The modularity of c implies that: $(c(D_1) + c(D_2)) = c(D_1 \cup D_2) - c(D_1 \cap D_2) = c(T_1 \cup T_2)$ (since the intersection of D_1 and D_2 is empty). This means that $\text{Utility}_1(\delta) + \text{Utility}_2(\delta) = c(T_1) + c(T_2) - c(T_1 \cup T_2) = c(T_1 \cap T_2)$.

The agents will agree on a deal that maximizes the product of the utilities (from the definition of a PMM, in Section 12). Since all pure deals have a constant sum of utility (which is $c(T_1 \cap T_2)$), the deals that maximize the product are those that have the smallest difference between the two agents' utilities. Therefore, there is either one pure deal, or two complementary pure deals, that maximize the product. In the former case, the unique pure deal gives both agents equal utility. In the latter case, one of the complementary pure deals will be chosen arbitrarily (each with probability $\frac{1}{2}$). Both agents will have the same *expected* utility ($\frac{1}{2}c(T_1 \cap T_2)$).

Assume that agent A_1 lies and creates a decoy task $d \in \mathcal{T}$. His apparent initial set of tasks is $T_1^* = T_1 \cup \{d\}$. Since $d \notin T_1$ (otherwise it was not a decoy task), $c(T_1^*) = c(T_1) + c(\{d\})$. For any pure deal $\delta = (D_1, D_2)$ the apparent utility for agent A_1 from the deal is: $c(T_1^*) - c(D_1)$; the actual utility that agent A_1 gets from δ depends on whether the decoy task d is in D_1. If $d \in D_1$, then A_1's actual utility from δ is $c(T_1) - (c(D_1) - c(\{d\})) = c(T_1) - c(D_1) + c(\{d\}) = c(T_1^*) - c(\{d\}) - c(D_1) + c(\{d\}) = c(T_1^*) - c(D_1)$ (i.e., A_1's actual utility from δ is equal to its apparent utility from δ).

If $d \notin D_1$ then its actual utility from δ is $c(T_1) - c(D_1) = c(T_1) - c(D_1) + c(\{d\}) = c(T_1^*) - c(D_1) - c(\{d\})$ (i.e., its actual utility from δ is lower than its apparent utility, by $c(\{d\})$).

There are two cases:

(1) $d \in T_2$. In this case, $c(T_1^* \cap T_2) = c(T_1 \cap T_2) + c(\{d\})$, and agent A_1's expected actual utility is $\frac{1}{2}c(T_1 \cap T_2)$. He would have gotten the same actual utility declaring his true set of tasks.

(2) $d \notin T_2$. Then $T_1^* \cap T_2 = T_1 \cap T_2$, and agent A_1's expected actual utility is $\frac{1}{2}(c(T_1 \cap T_2) - c(\{d\}))$. He would have gotten more $(\frac{1}{2}c(T_1 \cap T_2))$ declaring his true set of tasks. $\square$

Theorem 26. *For any encounter in a two-agent subadditive TOD, and any PMM over all-or-nothing deals, if agent A_1 knows that the cost of T_1 is greater than the cost of T_2, he can generate a default decoy lie that may benefit him, and will never harm him.*

Proof. Using Theorem 19, we know that agents will agree on a deal that gives them equal utility. Negotiating over all-or-nothing deals means that they will agree on a probability p that satisfies the following relationship: $c(T_1) - p * c(T_1 \cup T_2) = c(T_2) - (1-p) * c(T_1 \cup T_2)$. Thus, they will agree on the following: $p = (c(T_1) - c(T_2) + c(T_1 \cup T_2))/(2c(T_1 \cup T_2))$. If agent A_1 declares T_1^* which is a superset of T_1 (i.e., T_1 with extra decoy tasks), then they will agree on a p^* that satisfies $(c(T_1^*) - c(T_2) + c(T_1^* \cup T_2))/(2c(T_1^* \cup T_2))$. We have that $c(T_1^* \cup T_2) \geqslant c(T_1 \cup T_2)$ because of the monotonicity of c. If agent A_1 has chosen his decoys such that $c(T_1^*) = c(T_1)$, then we have that A_1's apparent utility is $c(T_1^*) - p^* * c(T_1^* \cup T_2)$, while his actual utility is $c(T_1) - p^* * c(T_1 \cup T_2)$ (since he won't have to carry out the decoy tasks). Thus, if $p^* < p$, agent A_1 benefits from this lie.

If $c(T_1) \geqslant c(T_2)$, then $p^* \leqslant p$, from the definition of p^* and p, the equivalence of $c(T_1^*)$ and $c(T_1)$, and the inequality $c(T_1^* \cup T_2) \geqslant c(T_1 \cup T_2)$. $\square$

References

[1] R. Axelrod, *The Evolution of Cooperation* (Basic Books, New York, 1984).
[2] K. Binmore, *Fun and Games, A Text on Game Theory* (Heath, Lexington, MA, 1992).
[3] S. Cammarata, D. McArthur and R. Steeb, Strategies of cooperation in distributed problem solving, in: *Proceedings IJCAI-83*, Karlsruhe (1983) 767-770.
[4] R. Clark, C. Grossner and T. Radhakrishnan, Consenses: a planning protocol for cooperating expert systems, in: *Proceedings Eleventh International Workshop on Distributed Artificial Intelligence*, Glen Arbor, MI (1992) 43-58.
[5] P.R. Cohen and H.J. Levesque, Intention = Choice + Commitment, in: *Proceedings AAAI-87*, Seattle, WA (1987) 410-415.
[6] P.R. Cohen and H.J. Levesque, Intention is choice with commitment, *Artif. Intell.* **42** (1990) 213-261.
[7] P.R. Cohen and H.J. Levesque, Teamwork, Technote 503, SRI International, Menlo Park, CA (1991).
[8] S.E. Conry, R.A. Meyer and V.R. Lesser, Multistage negotiation in distributed planning, in: A. Bond and L. Gasser, eds., *Readings in Distributed Artificial Intelligence* (Morgan Kaufmann, San Mateo, CA, 1988) 367-384.
[9] D.D. Corkill, Hierarchical planning in a distributed environment, in: *Proceedings IJCAI-79*, Tokyo (1979) 168-175.
[10] D.D. Corkill, A framework for organizational self-design in distributed problem solving networks, Ph.D. Thesis, University of Massachusetts, Amherst, MA (1982).
[11] K.S. Decker and V.R. Lesser, Generalizing the partial global planning algorithm, *Internat. J. Intell. Coop. Inform. Syst.* **1** (1992) 319-346.
[12] J. Doyle, Rationality and its roles in reasoning, *Comput. Intell.* **8** (1992) 376-409.
[13] E.H. Durfee, *Coordination of Distributed Problem Solvers* (Kluwer Academic Publishers, Boston, MA, 1988).
[14] E.H. Durfee and V.R. Lesser, Using partial global plans to coordinate distributed problem solvers, in: *Proceedings IJCAI-87*, Milan (1987) 875-883.

[15] E.H. Durfee, V.R. Lesser and D.D. Corkill, Cooperation through communication in a distributed problem solving network, in: M.N. Huhns, ed., *Distributed Artificial Intelligence* (Morgan Kaufmann, Los Altos, CA, 1987) Chapter 2, 29–58.

[16] E.H. Durfee and J.S. Rosenschein, Distributed problem solving and multi-agent systems: comparisons and examples, in: *Proceedings Thirteenth International Workshop on Distributed Artificial Intelligence*, Seattle, WA (1994) 94–104.

[17] E. Ephrati, Divide and conquer in multi-agent planning, in: *Proceedings AAAI-94*, Seattle, WA (1994) 375–380.

[18] E. Ephrati and J.S. Rosenschein, The Clarke Tax as a consensus mechanism among automated agents, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 173–178.

[19] E. Ephrati and J.S. Rosenschein, Constrained intelligent action: planning under the influence of a master agent, in: *Proceedings AAAI-92*, San Jose, CA (1992) 263–268.

[20] E. Ephrati and J.S. Rosenschein, Planning to please: planning while constrained by a master agent, in: *Proceedings Eleventh International Workshop on Distributed Artificial Intelligence*, Glen Arbor, MI (1992) 77–94.

[21] E. Ephrati and J.S. Rosenschein, Reaching agreement through partial revelation of preferences, in: *Proceedings Tenth European Conference on Artificial Intelligence*, Vienna (1992) 229–233.

[22] E. Ephrati and J.S. Rosenschein, Distributed consensus mechanisms for self-interested heterogeneous agents, in: *Proceedings First International Conference on Intelligent and Cooperative Information Systems*, Rotterdam (1993) 71–79.

[23] E. Ephrati and J.S. Rosenschein, Multi-agent planning as a dynamic search for social consensus, in: *Proceedings IJCAI-93*, Chambery (1993) 423–429.

[24] E. Ephrati and J.S. Rosenschein, Deriving consensus in multi-agent systems, *Artif. Intell.* **87** (1996).

[25] D. Fudenberg and J. Tirole, *Game Theory* (MIT Press, Cambridge, MA, 1992).

[26] L. Gasser, Social conceptions of knowledge and action: DAI foundations and open systems semantics, *Artif. Intell.* **47** (1991) 107–138.

[27] L. Gasser, Social knowledge and social action, in: *Proceedings IJCAI-93*, Chambery (1993) 751–757.

[28] M.P. Georgeff, Communication and interaction in multi-agent planning, in: *Proceedings AAAI-83*, Washington, DC (1983) 125–129.

[29] M.P. Georgeff, A theory of action for multi-agent planning, in: *Proceedings AAAI-84*, Austin, TX (1984) 121–125.

[30] M.P. Georgeff and A.L. Lansky, Reactive reasoning and planning, in: *Proceedings AAAI-87*, Seattle, WA (1987) 677–682.

[31] B.J. Grosz and S. Kraus, Collaborative plans for group activities, in: *Proceedings IJCAI-93*, Chambery (1993) 367–373.

[32] J.C. Harsanyi, Approaches to the bargaining problem before and after the theory of games: a critical discussion of Zeuthen's, Hick's and Nash theories, *Econometrica* **24** (1956) 144–157.

[33] J.C. Harsanyi, *Rational Behavior and Bargaining Equilibrium in Games and Social Situations* (Cambridge University Press, Cambridge, 1977).

[34] M. Kamel and A. Syed, An object-oriented multiple agent planning system, in: L. Gasser and M.N. Huhns, eds., *Distributed Artificial Intelligence, Volume* II (Pitman, London/Morgan Kaufmann, San Mateo, CA, 1989) 259–290.

[35] M.J. Katz and J.S. Rosenschein, Verifying plans for multiple agents, *J. Exper. Theoret. Artif. Intell.* **5** (1993) 39–56.

[36] D. Kinny, M. Ljungberg, A. Rao, E. Sonenberg, G. Tidhar and E. Werner, Planned team activity, in: *Pre-Proceedings Fourth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*, Rome (1992) Chapter 2.

[37] K. Konolige, A first-order formalization of knowledge and action for a multi-agent planning system, *Mach. Intell.* **10** (1982).

[38] S. Kraus and J. Wilkenfeld, Negotiations over time in a multi-agent environment: preliminary report, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 56–61.

[39] S. Kraus, J. Wilkenfeld and G. Zlotkin, Multiagent negotiation under time constraints, *Artif. Intell.* **75** (1995) 297–345.

[40] T. Kreifelts and F. von Martial, A negotiation framework for autonomous agents, in: Y. Demazeau and J.-P. Müller, eds., *Decentralized A.I.2, Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (North-Holland, Amsterdam, 1991) 71-88.

[41] K. Kuwabara and V.R. Lesser, Extended protocol for multistage negotiation, in: *Proceedings Ninth Workshop on Distributed Artificial Intelligence*, Rosario, WA (1989) 129-161.

[42] B. Lâasri, H. Lâasri and V.R. Lesser, Negotiation and its role in cooperative distributed problem solving, in: *Proceedings Tenth International Workshop on Distributed Artificial Intelligence*, Austin, TX (1990) Chapter 8.

[43] R.D. Luce and H. Raiffa, *Games and Decisions* (Wiley, New York, 1957).

[44] T.W. Malone, R.E. Fikes, K.R. Grant and M.T. Howard, Enterprise: a market-like task scheduler for distributed computing environments, in: B.A. Huberman, ed., *The Ecology of Computation* (North-Holland, Amsterdam, 1988).

[45] D. McArthur, R. Steeb and S. Cammarata, A framework for distributed problem solving, in: *Proceedings AAAI-82*, Pittsburgh, PA (1982) 181-184.

[46] L. Morgenstern, A first order theory of planning, knowledge, and action, in: J.Y. Halpern, ed., *Theoretical Aspects of Reasoning About Knowledge* (Morgan Kaufmann, Los Altos, CA, 1986) 99-114.

[47] L. Morgenstern, A formal theory of multiple agent nonmonotonic reasoning, in: *Proceedings AAAI-90*, Boston, MA (1990) 538-544.

[48] Y. Moses and M. Tennenholtz, Artificial social systems, Part 1: basic principles, Tech. Rept. CS90-12, Weizmann Institute of Sciences, Rehovot, Israel (1990).

[49] Y. Moses and M. Tennenholtz, On computational aspects of artificial social systems, in: *Proceedings Eleventh International Workshop on Distributed Artificial Intelligence*, Glen Arbor, MI (1992) 267-283.

[50] J.F. Nash, The bargaining problem, *Econometrica* **28** (1950) 155-162.

[51] E.P.D. Pednault, Formulating multiagent dynamic-world problems in the classical planning framework, in: M.P. Georgeff and A.L. Lansky, eds., *Reasoning about Actions and Plans: Proceedings of the 1986 Workshop* (Morgan Kaufmann, San Mateo, CA, 1987) 47-82.

[52] M.E. Pollack and M. Ringuette, Introducing the Tileworld: experimentally evaluating agent architectures, in: *Proceedings AAAI-90*, Boston, MA (1990) 183-189.

[53] R.P. Pope, S.E. Conry and R.A. Mayer, Distributing the planning process in a dynamic environment, in: *Proceedings Eleventh International Workshop on Distributed Artificial Intelligence*, Glen Arbor, MI (1992) 317-331.

[54] A.S. Rao, M.P. Georgeff and E.A. Sonenberg, Social plans: a preliminary report, in: *Pre-Proceedings Third European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds*, Germany (1991).

[55] J.S. Rosenschein, Synchronization of multi-agent plans, in: *Proceedings AAAI-82*, Pittsburgh, PA (1982) 115-119.

[56] J.S. Rosenschein, Rational interaction: cooperation among intelligent agents, Ph.D. Thesis, Stanford University, Stanford, CA (1986).

[57] J.S. Rosenschein, Consenting agents: Negotiation mechanisms for multi-agent systems, in: *Proceedings IJCAI-93*, Chambery (1993) 792-799.

[58] J.S. Rosenschein and M.R. Genesereth, Deals among rational agents, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 91-99.

[59] J.S. Rosenschein and G. Zlotkin, *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers* (MIT Press, Cambridge, MA, 1994).

[60] A.E. Roth, *Axiomatic Models of Bargaining* (Springer, Berlin, 1979).

[61] Y. Shoham and M. Tennenholtz, Emergent conventions in multi-agent systems: initial experimental results and observations (preliminary report), in: *Proceedings Third International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA (1992).

[62] Y. Shoham and M. Tennenholtz, On the synthesis of useful social laws for artificial agent societies (preliminary report), in: *Proceedings AAAI-92*, San Jose, CA (1992).

[63] R.G. Smith, A framework for problem solving in a distributed processing environment, Ph.D. Thesis, Stanford University, Stanford, CA (1978).

[64] R.G. Smith, The contract net protocol: high-level communication and control in a distributed problem solver, *IEEE Trans. Comput.* **29** (1980) 1104-1113.

[65] R. Steeb, S. Cammarata, F. Hayes-Roth and R. Wesson, Distributed intelligence for air fleet control, Tech. Rept. WD-839-ARPA, The Rand Corporation (1980).

[66] C.J. Stuart, An implementation of a multi-agent plan synchronizer, in: *Proceedings IJCAI-85*, Los Angeles, CA (1985) 1031–1035.

[67] K.P. Sycara, Resolving goal conflicts via negotiation, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 245–250.

[68] K.P. Sycara, Argumentation: planning other agents' plans, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 517–523.

[69] F. von Martial, Multiagent plan relationships, in: *Proceedings Ninth International Workshop on Distributed Artificial Intelligence*, Rosario, WA (1989) 59–72.

[70] F. von Martial, Coordination of plans in multiagent worlds by taking advantage of the favor relation, in: *Proceedings Tenth International Workshop on Distributed Artificial Intelligence*, Austin, TX (1990) Chapter 21.

[71] F. von Martial, Coordination by negotiation based on a connection of dialogue states with actions, in: *Proceedings Eleventh International Workshop on Distributed Artificial Intelligence*, Glen Arbor, MI (1992) 227–246.

[72] M.P. Wellman, A general equilibrium approach to distributed transportation planning, in: *Proceedings AAAI-92*, San Jose, CA (1992).

[73] M.P. Wellman, A market-oriented programming environment and its application to distributed multicommodity flow problems, *J. Artif. Intell. Res.* **1** (1993) 1–23.

[74] F. Zeuthen, *Problems of Monopoly and Economic Walfare* (G. Routledge & Sons, London, 1930).

[75] G. Zlotkin and J.S. Rosenschein, Negotiation and task sharing among autonomous agents in cooperative domains, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 912–917.

[76] G. Zlotkin and J.S. Rosenschein, Negotiation and conflict resolution in non-cooperative domains, in: *Proceedings AAAI-90*, Boston, MA (1990) 100–105.

[77] G. Zlotkin and J.S. Rosenschein, Cooperation and conflict resolution via negotiation among autonomous agents in noncooperative domains, *IEEE Trans. Syst. Man Cybern.* **21** (1991) 1317–1324.

[78] G. Zlotkin and J.S. Rosenschein, Incomplete information and deception in multi-agent negotiation, in: *Proceedings IJCAI-91*, Sydney, Australia (1991) 225–231.

[79] G. Zlotkin and J.S. Rosenschein, Negotiation and goal relaxation, in: Y. Demazeau and J.P. Müller, eds., *Decentralized A.I.2, Proceedings of the Second European Workshop on Modelling Autonomous Agents in a Multi-Agent World* (North-Holland, Amsterdam, 1991) 273–286.

[80] G. Zlotkin and J.S. Rosenschein, Compromise in negotiation: exploiting worth functions over states, Tech. Rept. 93-3, Leibniz Center for Computer Science, Hebrew University, Jerusalem (1993).

[81] G. Zlotkin and J.S. Rosenschein, A domain theory for task oriented negotiation, in: *Proceedings IJCAI-93*, Chambery (1993) 416–422.

[82] G. Zlotkin and J.S. Rosenschein, The extent of cooperation in state-oriented domains: negotiation among tidy agents, *Comput. Artif. Intell.* **12** (1993) 105–122.

[83] G. Zlotkin and J.S. Rosenschein, Negotiation with incomplete information about worth: strict versus tolerant mechanisms, in: *Proceedings First International Conference on Intelligent and Cooperative Information Systems*, Rotterdam (1993) 175–184.

[84] G. Zlotkin and J.S. Rosenschein, Mechanisms for automated negotiation in state oriented domains, *J. Artif. Intell. Res.* (to appear).

[85] G. Zlotkin and J.S. Rosenschein, Compromise in negotiation: exploiting worth functions over states, *Artif. Intell.* **84** (1996) 151–176.