Consenting Agents: Negotiation Mechanisms for Multi-Agent Systems

Jeffrey S, Rosenschein*
Computer Science Department
Hebrew University
Givat Ram, Jerusalem, Israel

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 fields such as decision theory and 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.

1 Machines Controlling and Sharing Resources

Computers are making more and more decisions in a relatively autonomous fashion. Telecommunications 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.

*This research has been partially supported by the Israeli Ministry of Science and Technology (Grant 032-8284).

2 Heterogeneous, Self-motivated Agents

In the field of distributed artificial intelligence (DAI), researchers explore methods that enable the coherent interaction of computers in distributed systems. One of the major distinctions in DAI is between research in Distributed Problem Solving (DPS) [Smith, 1978; Conry *et al.*, 1988; Durfee, 1988; Clark *et* al., 1992], in which the distributed system has been centrally designed, and Multi-Agent (MA) Systems [Kraus and Wilkenfeld, 1991; Ephrati and Rosenschein, 1991; Kreifelts and von Martial, 1990; Sycara, 1989], 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 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 [Doyle, 1992]), acts accordingly to increase that welfare.

The approach of MA system research is particularly appropriate for the kinds of scenarios mentioned above. When the 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" [Gasser, 1991], 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 multi-agent systems is thus the appropriate model with which to analyze these independent software agents and their interactions.

3 The Aim of the Research

The purpose of the research described in this paper 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. The development of "social laws" has parallels with the work of [Shoham and Tennenholtz, 1992]. 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 inter-agent 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 who 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 [Fudenberg and Tirole, 1991].

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 Optimally (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 agent should have an incentive to deviate from agreed-upon strategies. The strategy that agents adopt 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: We may not want agents to play different roles in the interaction scenario. This simplifies the overall mechanism, and removes the question of which agent will play which role when an interaction gets under way.

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 [Zlotkin and Rosenschein, 1993b]). 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.

6 Domain Theory

I 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.

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, in a paper [Zlotkin and Rosenschein, 1993a] that appears in these proceedings). 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 $30,000 a year," while another agent may want the records of "All female employees with more than three children." Both agents share a sub-task, 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.

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 [Zlotkin and Rosenschein, 1990]. 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 (WOD) 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 The World, as discussed in [Pollack and Ringuette, 1990]. 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).

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.

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 in [Zlotkin and Rosenschein, 1993a].

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. In order 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 sub-queries, 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 [Zlotkin and Rosenschein, 1991]. 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 Multi-Agent 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 Multi-Agent systems, in considering efficiency without considering stability.[1] Without stability, efficiency cannot be guaranteed, as agents are tempted to deviate from the efficient strategy.

[1]Though in Distributed Problem Solving systems, where there is a central designer of all the agents, stability need not be a serious issue [Shoham and Tennenholtz, 1992].

10 Conclusions

Computers are making an increasing number of decisions autonomously, and interacting machines, capable of reaching mutually beneficial agreements, have an important role to play in daily life. The field of distributed artificial intelligence, and particularly its subfield of multi-agent systems, provides an appropriate model for studying these systems of heterogeneous, self-motivated agents.

To provide the agents with a suitable interaction environment in which they can coordinate, it is necessary to establish high-level protocols, that motivate socially beneficial (and individually beneficial) behavior. Game theory can provide tools appropriate to the design of these distributed systems. 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.

Acknowledgments

The research issues described in this paper have been developed over a period of several years in close collaboration with Gilad Zlotkin.

References

[Clark *et al,* 1992] R. Clark, C. Grossner, and T. Radhakrishnan. Consensus: A planning protocol for cooperating expert systems. In *Proceedings of the Eleventh International Workshop on Distributed Artificial Intelligence,* pages 43-58, Glen Arbor, Michigan, February 1992.

[Conry *et al,* 1988] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence,* pages 367-384. Morgan Kaufmann Publishers, Inc., San Mateo, California, 1988.

[Doyle, 1992] Jon Doyle. Rationality and its roles in reasoning. *Computational Intelligence,* 8(2):376-409, May 1992.

[Durfee, 1988] Edmund H. Durfee. *Coordination of Distributed Problem Solvers.* Kluwer Academic Publishers, Boston, 1988.

[Ephrati and Rosenschein, 1991] Eithan Ephrati and Jeffrey S. Rosenschein. The Clarke Tax as a consensus mechanism among automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence,* Anaheim, California, July 1991.

[Fudenberg and Tirole, 1991] Drew Fudenberg and Jean Tirole. *Game Theory.* The MIT Press, Cambridge, Massachusetts, 1991.

[Gasser, 1991] Les Gasser. Social conceptions of knowledge and action: DAI foundations and open systems semantics. Artificial Intelligence, 47(1-3):107-138 1991.

[Kraus and Wilkenfeld, 1991] Sarit Kraus and Jonathan Wilkenfeld. Negotiations over time in a multi agent environment: Preliminary report. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pages 56-61, Sydney, Australia, August 1991.

[Kreifelts and von Martial, 1990] Thomas Kreifelts and Frank von Martial. A negotiation framework for autonomous agents. In Proceedings of the Second European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds, pages 169-182, Saint-Quentin en Yvelines, France, August 1990.

[Pollack and Ringuette, 1990] Martha E. Pollack and Marc Ringuette. Introducing the tileworld: Experimentally evaluating agent architectures. In Proceedings of the National Conference on Artificial Intelligence, pages 183-189, Boston, Massachusetts, August 1990.

[Shoham and Tennenholtz, 1992] Yoav Shoham and Moshe Tennenholtz. On the synthesis of useful social laws for artificial agent societies (preliminary report). In Proceedings of the Tenth National Conference on Artificial Intelligence, pages 276-281, San Jose, July 1992.

[Smith, 1978] Reid G. Smith. A Framework for Problem Solving in a Distributed Processing Environment. PhD thesis, Stanford University, 1978.

[Sycara, 1989] Katia P. Sycara. Argumentation: Planning other agents' plans. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, pages 517-523, Detroit, Michigan, August 1989.

[Zlotkin and Rosenschein, 1990] Gilad Zlotkin and Jeffrey S. Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In Proceedings of the Eighth National Conference on Artificial Intelligence, pages 100-105, Boston, Massachusetts, July 1990.

[Zlotkin and Rosenschein, 1991] Gilad Zlotkin and Jeffrey S. Rosenschein. Incomplete information and deception in multi-agent negotiation. In Proceedings of the Twelfth International Joint Conference on Artificial Intelligence, pages 225-231, Sydney, Australia, August 1991.

[Zlotkin and Rosenschein, 1993a] Gilad Zlotkin and Jeffrey S. Rosenschein. A domain theory for task oriented negotiation. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, Chambery, France, August 1993.

[Zlotkin and Rosenschein, 1993b] Gilad Zlotkin and Jeffrey S. Rosenschein. Negotiation with incomplete information about worth: Strict versus tolerant mechanisms. In Proceedings of the First International Conference on Intelligent and Cooperative Information Systems, Rotterdam, The Netherlands, May 1993. To appear.