

Policing Virtual Organizations

Nir Oren, Tim Norman, Alun Preece, Stuart Chalmers

Abstract

Autonomous agents within a multi-agent system can undertake behaviors that, while profitable for them, are damaging to the system as a whole. One way of regulating an agent's behavior is by obliging it to act in accordance with a set of specifications set out in a contract, and sanctioning it otherwise. In this document, We propose a framework that performs two major subtasks of contract based behavior enforcement in partially observable multi-agent environments: behavior specification and behavior violation detection. The framework consists of three main components. The first is a contract specification language used to specify desired agent behavior. The second is a dialogue game used to present and refute evidence about an agent's actual behavior. The final component of the framework is a mechanism used to combine the evidence so as to reach a decision regarding an agent's guilt or innocence. We also examine existing approaches for performing contract enforcement, and describe a concrete environment where investigations into the effects of policing can take place.

1 Introduction

A group of autonomous software agents may temporarily pool resources to form a virtual organization (VO) so as to achieve goals that they would be unable to individually, or provide services more efficiently (for example, at a lower cost to themselves) than they would be able to alone. While it is assumed that the agents forming the VO are self interested, each agent stands to gain by participating in the VO. For example, a group of agents could pool their resources to bid for (and then service) a contract for which they would not have the individual capacity to fulfill.

The problem of VO formation is discussed in detail in [9]. Once a VO is formed, it will act as a single cohesive unit in the context of service provision.

Given an environment in which VO formation can occur, an agent can respond to a request for its services in a number of ways:

1. Ignore the request.
2. Fulfill the request as an individual.
3. Honor the request using resources from one of the VOs it represents.
4. Attempt to form a new VO to satisfy the request.
5. Break existing VO commitments to satisfy the request.

Clearly, agents require relatively complex decision-making capabilities to determine which of these actions to take. An additional complication arises due to the assumption that agents have only a finite amount of resources that they may commit during any time-period. Agents may thus often end up reneging on their existing commitments so that they may take part in more profitable ventures. While this behavior is profitable for the individual agent, it is harmful for the environment as a whole; very few clients would be willing to use a marketplace wherein traders frequently fail to deliver on their promises.

This paper discusses techniques for failure detection and blame assignment in the context of VO based environments. The work is situated within the multi-site CONOISE-G project ¹ in which we are investigating

¹CONOISE-G in turn is built on top of the framework constructed for the CONOISE project, more details of which are available at <http://www.conoise.org>

multiple issues relating to reliable service delivery in marketplaces containing VOs. Our research is highly integrated with the CONOISE test-bed multi-agent system. Thus, before examining policing in such an environment, we first situate the problem within a scenario and describe the test-bed architecture.

2 The Scenario and Environment

This section begins by describing the domain in which our research is situated. The scenario is sketched out, after which the various agents operating in the environment are examined.

The domain consists of a number of service provider agents (SPs), each of which is able to provide a number of multimedia services to a set of clients. Currently defined services include text-messaging, news clippings, streamed movies and phone calls. Each SP has a limited number of resources it can provide at any one time. Client agents represent the user, and are able to request services from the SPs. The environment provides a number of infrastructure services, instantiated as agents:

- Yellow pages agents (YPs) store SP contact details, as well as information concerning the services advertised by SPs.
- A clearing agent (CA) allows an agent to determine which set of bids best fulfills its requirements, based on the agent's request and various parameters associated with each bid (such as the bid price).
- A quality agent (QA) is able to assign a quality rating to the services offered by an agent. This quality rating can be used by the CA to determine the overall utility of an agent's bid.
- A monitoring agent abstracts the various sensors in the system, using a publish-subscribe pattern. Agents interested in observing the environment contact the monitoring agent and tell it what services (and which service properties) should be monitored, and for what type of events. When such an event occurs, the monitoring agent sends a message to the interested agent, informing it of the event.
- A trust agent models the levels of trust between the various agents in the system. The trust value can be used as one of the parameters passed to the CA so as to allow it to make a decision regarding which bids should be accepted.
- The policing agent, described in more detail in Section 3, is contacted by agents when they believe a breach of contract has occurred. This agent then initiates an investigation, gathers evidence, and reports whether a breach of contract has occurred, as well as which agent breached the contract, to the various agents in the system. These results can then be used to update the trust system, thereby reducing the likelihood of irresponsible (or malicious) agents obtaining contracts. Output from the policing agent can also be used to institute sanctions against a misbehaving agent.

Within our system, a provider agent can also take on the role of a VO manager (VOM). An agent in the VOM role represents the entire VO in interactions with other agents.

A number of distinct phases of system operation can be defined. The first, VO formation, occurs in response to a client's request for a service. The second phase is VO operation, wherein a client makes use of the services of a VO. VO termination occurs once a VO has reached the end of its useful life and is terminated in an orderly manner, while VO perturbations occur when an unexpected event (such as an uncooperative agent) arises and causes the VO to reform. We will now examine the various phases in more detail.

2.1 VO Formation

The interaction between the various agents in a typical VO formation process is shown in Figure 1. The formation episode is initiated by a request from the client to a provider agent. This SP adopts the VOM role, and queries the YP for a list of agents able to satisfy the user's query. After obtaining a response from the YP, the VOM sends the appropriate SPs a bid request message. Any bids received are then collated

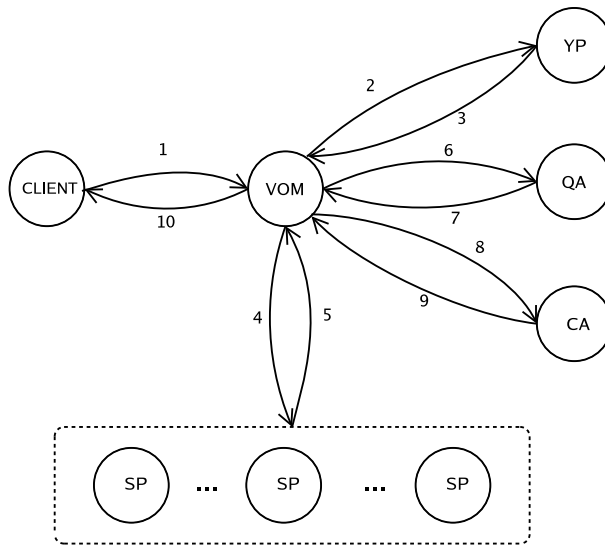


Figure 1: The agents making up the VO and the environment, together with the order of messages passed between them during a VO formation episode.

and forwarded to the QA, which assigns a quality rating to each bid component. The combined bids and quality ratings are then sent to the CA, which returns a set of winning bids to the VOM. Successful SPs are informed regarding which services they are to provide, and the client is notified regarding which services the VO is able to fulfill for it.

The request for services sent by the VOM to the SPs can be viewed as a form of contract. This request contains parameters such as minimum service levels, and can thus be used to determine whether an agent has violated its obligations by providing an inferior service. Further refinements of the CONOISE-G framework may include more complex contract negotiation mechanisms, as well as a richer contract specification language.

2.2 VO Operation

After a VO has formed, it must provide its services to the client. Since VO abstraction² is an important concept, a service request from a client is routed through the VOM to the appropriate SP. Since our architecture allows for VOs to supply services to other VOs, the message may pass through a number of VOMs before reaching the appropriate SP. VOMs may need to make some decisions during the message routing phase so as to best use the resources at their disposal. Once a SP has received a service request, it can begin fulfilling it. In our testbed, service fulfillment is simulated outside the agent environment. This design decision was taken so as to allow us to simulate arbitrary services, ranging from Grid based services, to web services. Such a design even allows us to simulate real world service delivery.

2.3 VO Perturbation

This phase of operation can arise due to a number of events in the environment, including, but not limited to:

- A SP not fulfilling its obligations

²i.e. the client not knowing whether it is getting its services from a VO or from a single agent

- The VOM deciding to reorganize the VO to take advantage of the environment (e.g. when a new, very cheap SP appears).
- A communications failure removing some of the SPs from the environment.
- The VO having fulfilled all its commitments.

When a perturbation occurs, the VO is forced to reform or terminate. If the VOM decides that VO reformation should occur, it determines which services must be replaced, and then triggers a new VO formation episode by sending itself a request for these services. The VO generated by the formation episode is combined with the existing VO.

If the VOM decides that the VO should cease to exist, it sends a message telling all VO members this, allowing them to free up any resources they have committed to the VO.

The environment was designed to be FIPA³ compliant, and our implementation makes use of the JADE⁴ and Zeus⁵ agent platforms. Agents communicate with each other by exchanging FIPA ACL messages. The content of these messages is defined using lightweight ontologies expressed using Semantic Web representations⁶. It should be noted that agents are designed to be able to operate within a new domain by a simple modification of some of these ontologies.

3 Policing Mechanisms

This section describes our thoughts on methods for developing a policing framework for open multi-agent systems containing complex social structures such as VOs. As mentioned previously, agents have the ability to renege on commitments. To prevent this from happening, the contracts signed by the agents contain sanctions. Furthermore, when an agent provides an inferior service (or breaks its commitments), its quality and trust ratings may drop, leading to a smaller chance of being selected for further service provision. Detecting violations in contracts, and blaming the correct agent (or agents) is clearly of vital importance in such an environment.

Currently, SPs use a constraint satisfaction programming technique [2] to maximize the amount of profit they make by allocating their (limited) resources in an optimal way. Since the trust and sanctioning mechanisms are still to be integrated into the system, agents currently break commitments with no second thoughts. Furthermore, the environment is currently fully observable. Detecting which agents have reneged on their commitments thus consists of parsing the contract, matching parts of the environment to the relevant contract clauses, and then checking if they are in a valid state. While not trivial, this is a reasonably simple task.

To allow for an investigation into policing in VO based marketplaces, we have begun extending our environment in a number of ways. Infrastructure agents such as the monitoring agent have been implemented to allow us to easily present different views of the environment to different agents. The trust component is able to provide agents with a subjective view of other agents reputations and allows for the implementation of non-monetary sanctions.

Contracts between agents are another extension to the CONOISE framework. While contracts are currently implicitly signed between agents during the VO formation process, we intend to extend our framework to cater for explicit contracts using the WS-SLA (Web Services Service Level Agreement) standard. This will allow us to specify what service levels each agent should meet when providing services. SPs (and client agents) are currently too simple to automatically generate complete contracts based on their bids (as complex negotiation would be required to decide on various contract elements such as sanctions and acceptable service levels). Thus, we intend to provide agents with a set of contract templates; the VOM then “fills

³<http://www.fipa.org>

⁴<http://sharon.cselt.it/projects/jade>

⁵<http://more.btexact.com/projects/agents/zeus>

⁶RDF(s)

in the blanks” when an agreement to provide (and consume) services is reached, and passes the completed contracts onto the client and SPs [7].

Extending the CONOISE environment to allow for contracts and the partial observability of states will let us begin an investigation into policing in such domains.

3.1 Agent Design

Policing in partially observable domains containing rich social structures requires reasoning about a number of different things. First, it must be able to determine what should have occurred. This is specified using a contracting language. Second, a method for gathering evidence to determine what actually happened is needed. Finally, since the evidence gathering mechanism may yield conflicting evidence, a method for combining the gathered evidence to reach a final conclusion is required.

We intend to attack the problem by following a path similar to how humans handle such situations: the policing agent will gather what evidence it can, and then initiate a dialogue with other agents in the system to gather further evidence. The accusing agent (plaintiff) and accused (defendant) are further allowed to submit arguments and counter-arguments regarding the evidence, in an attempt to prove their guilt (or innocence). The policing agent will then act as a judge, reasoning over the submitted evidence and arguments to reach a final conclusion determining which (if any) agents are guilty of the infraction. Our framework is not responsible for the administration of sanctions.

In the remainder of this section, we will discuss each component of such a policing framework in more detail.

3.1.1 The Contracting Language

Our contract representation language should allow for the representation of standard contract constructs, including temporal deadlines, actions, service levels, norms and sanctions. Due to our goal of operating within rich social environments, additional concepts such as group action and responsibility as well as the transfer of norms between agents should also be representable. Furthermore, our language should have well defined semantics (based on a logic) so that no confusion can arise regarding a contract’s meaning. We intend to extend LCR [4] to allow for these additional constructs and provide us with well defined formal semantics.

3.1.2 The Dialogue Game

The dialogue game is used by the framework to gather evidence regarding the state of the unobservable parts of the environment. A dialogue game based framework requires a number of components to be defined:

- A language in which the participants can make statements.
- Contextual rules defining which statements are allowed to be uttered at each stage of the dialogue.
- Rules defining the effects of an agent’s utterances on the dialogue game participant’s knowledge structures. For example many dialogue games have a rule stating that if an agent makes an utterance that *A* is the case, then all dialogue game participants should add the fact that the agent is committed to the fact that *A* is true to their knowledge stores.
- Rules stating how and when a dialogue may be initiated, as well as when a dialogue is terminated are also considered to be part of the dialogue game, though many researchers do not focus on these types of rules.

These rules are sufficient to determine whether an agent follows the dialogue game protocol or not. However, they give no guidance regarding which locutions an agent should make in the course of a dialogue game. To take part in our proposed dialogue game for reasoning about evidence gathering, agents need

to be able to determine what statements they should utter next, and what evidence they should present. Techniques for doing this range from using heuristics, to planning, to theorem proving.

The concept of burden of proof exists in human legal argumentation. When a dialogue participant presents some evidence in a case, it may be his responsibility to provide further evidence before it is accepted as a fact. Alternatively, the evidence may be accepted by default, and the burden of proof would then fall on the other party arguing against the agent's position to disprove the evidence. Normally, responsibility for assigning the burden of proof lies in the hands of a judge. In our system, the policing framework must be able to reason regarding the assignment of the burden of proof, and communicate it to the other dialogue participants as appropriate.

3.1.3 Evidence Fusion

Once all evidence has been gathered (i.e. no agent puts forward any new arguments), the policing agent must make a decision, based on the presented evidence and arguments, regarding the guilt or innocence of the defendant. As briefly discussed in the next section, multiple techniques for combining evidence to reach a decision exist. We intend to have a model for the determination of guilt, taking into account the strength of arguments based on both the type of argument and the strength of evidence for (or against) it, as well as the strength of its supporting and undercutting arguments.

3.2 Background and Related Research

Conceptually, the tasks of gathering evidence and combining it to form a conclusion can be viewed separately. Most researchers appear to ignore the separability of these concepts however, and deal with both problems simultaneously.

It is possible to recast contract execution enforcement as a sensing problem with byzantine sensors. When viewed this way, work done in the field using probability and uncertainty theory can be brought to bear on the problem. Daskalopulu et al.[3] have approached the problem this way by suggesting an approach utilizing subjective logic[6]. Subjective logic is based on Dempster–Schafer uncertainty theory. In their framework, the agent's belief in the state of the environment is combined with a set of reputation ratings in an effort to determine the most likely real environment state. While this approach works in many situations, it fails in richer environments with more complex agents. For example, an agent who always reported honestly until now may provide false information due to a conflict of interests. Since its information would be highly trusted, it is very likely that an incorrect conclusion would be reached. Furthermore, failures due to circumstances outside the agent's control, agent collusion and sabotage by other agents will often cause such a system to blame an incorrect set of agents.

Rather than addressing the problem of determining the actual state of the world, many researchers assume a fully-observable environment, and concern themselves with creating a sufficiently well defined contract representation so that agents will have no semantic "wiggle-room" if a contract violation occurs. LCR for example, specifies a branching time deontic action logic explicitly containing the concept of a violation. Given such a contract representation, and a fully observable environment, a monitoring agent would (possibly using a theorem prover) easily be able to detect the violation of any obligations by agents. Many other researchers have investigated contract representation formalisms [1] [14]. Currently, no formal contract representation language currently exists that is able to deal with many of the artifacts found in relatively simple real world contracts. While many ad-hoc contracting languages exist (see for example [5] and [7]), their lack of formal semantics makes it difficult to determine when an agent has failed to fulfill its obligations.

As mentioned above, we have decided to attack the problem in a different manner, by using an argumentation based approach. A body of work already exists regarding knowledge discovery using argumentation. McBurney and Parsons have identified and formalized a dialogue for chance discovery in domains containing distributed knowledge [8]. This type of work is applicable to our research as they propose a dialogue game allowing agents to argue about the validity of evidence. Prakken has written a number of articles dealing with dialogue games and argumentation systems for legal reasoning, focusing on issues such as burden of

proof [12] and reasoning about evidence [11]. His approach is very similar to the research we intend to pursue, but our environment and goals differ from his in a number of significant ways:

1. Prakken's system assumes a neutral judge, passive judge. In our system, the policing agent may actively question other agents in an attempt to gather evidence.
2. His system lets agents submit evidence and attempt to refute each other's arguments. The policing agent actively interacts with the other agents in the system, requesting evidence from them, questioning them, and being questioned by them.
3. Due to the neutrality of the judge, the burden of proof is never on him in Prakken's system. Since our system interacts with other agents, it may be questioned and required to prove things itself.
4. Perhaps most importantly, Prakken offers a dialogue game and argumentation framework for legalistic arguments, but does not suggest how a computer system may reason using the system (i.e. at what he calls the "heuristic layer). Nor does he make any suggestions about how the judge should decide on whom the burden of proof falls. Our system clearly requires these details to be implemented.

Reed and Rowe [13] have described a charting tool which can aid a human judge in weighing up arguments and evidence in court cases. Models for weighing up the strength of arguments in dialogue games can range from comparing the number of arguments for or against a conclusion, to elaborate weighting schemes based on the type of argument put forth by an agent [10].

Having briefly surveyed related research areas, we now move onto describing our approach in more detail.

4 Scenario

In this section, we present a typical contract failure in our mobile application domain, and demonstrate how it will be dealt with by our policing agent.

Within the sample dialogue, deeper indentation indicates a subargument which attempts to refute its parent.

Assume that client agent *A* has signed a contract with provider agent *B*. The contract states that *B* must provide *A* with a text messaging service. *A* claims that it tried sending a text to *C*, and *B* had not delivered the message. *A* requests that the policing agent undertake an investigation. The policing agent starts by making sure that the contract does in fact state that *B* is responsible for providing *A* with a text service to *C*. When this is found to be the case, the policing agent initiates a dialogue with *A* and *B*. It puts forward the claim that *B* is in breach of contract.

- *B* responds stating that there is no evidence that *A* attempted to set a text.
 - *A* provides the policing agent with a log of texts it claims to have sent.
 - *B* claims that *A*'s logs are falsified.
 - *A* requests that the policing agent contact the people *A* had texted according to the logs, as proof that its logs are valid.
 - The policing agent declines, pointing out that nothing stops *A* from adding a false log entry to otherwise valid logs. *A* concedes to this point.
- *A* points to the contents of the text, claiming that it is in its best interest to have sent it. No agent is equipped to read and understand the text contents however.

- *A* indicates that a conversation between it and *C* had taken place, and that *C* was expecting a response. The policing agent approaches *C*, and finds out that this was indeed the case. *B* concedes to this line of argument.
- *B* then points out that the contract requires a 75% text delivery rate from it.
 - *A* points out that delivery failures had occurred before with *B*, but it had not complained due to the above-mentioned 75% success rate. With this transmission however, the success rate dropped below the rate specified in the service level agreement. *B* concedes to this.
- *B* does however point out that the reason the messages were not transmitted was due to circumstances beyond its control: a network failure (which *B* argues is *force majeure*) caused the delivery failure.
 - The policing agent now needs to determine if a network failure was at the heart of *B*'s problems. It requests *B*'s transmission logs, and approaches other agents to see if they had communications problems at the times that *B* claimed the network was down. It finds that very few (if any) agents experienced difficulties.
 - *B* points out that the agents could have been on different networks.
 - The policing agent responds by admitting that it is unsure what networks the agents were on, but that it had polled a significant number of agents, and that the chances are good that at least a few of them were on the same network. *B* accepts this possibility.
- *B* claims that the message was not transmitted as *C*'s mobile-phone memory was full at the time. *A* was informed of the failure.
 - The policing agent asks *C* what the state of the mobile was when the message was sent. *C* is unsure.
- The policing agent points out that *B* had contradicted itself, first arguing that *A* had not attempted to send a message, and then conceding to that fact, arguing that it was not its fault that the message had not been delivered.
 - *B* responds that it had not initially looked into its logs due to the cost of retrieving the information, but had later done so. The policing agent accepts this line of reasoning.

After performing this investigation, the policing agent rules in favor of *B*, claiming that *A* has not conclusively proved that *B* had violated its contract.

5 Conclusions

In this paper, we outlined a framework for failure detection and blame assignment which can operate in partially observable environments containing complex social structures and dishonest agents. Our approach is based on the way humans handle such policing tasks. Starting with a contract to determine what should have happened, the framework tries to determine what actually occurred. This is achieved by having the system gather what evidence it can, and then asking the agents involved in the dispute to argue about the merits of their position. The agents then bring in evidence to support their position and refute their opponents position. The presentation of evidence may require other agents to be questioned, possibly dragging them into the investigation. When no more arguments are presented, the framework merges the arguments presented by the agents to reach a final conclusion regarding whether a breach of contract occurred, and who was responsible for the breach.

In the future, socially complex environments containing agents created by different organizations will become more common. Since no guarantee of correct agent behavior in such environments can be assumed,

policing is required to ensure that agents do not renege on their responsibilities. We believe that our research provides a promising approach to handling policing in such environments.

Acknowledgments

The CONOISE-G project is jointly funded by the DTI/EPSRC E-Science Core Programme and BT Exact – British Telecom’s research, technology and IT operations business.

The project is a collaboration between BT Exact, and Aberdeen, Cardiff and Southampton Universities.

References

- [1] A.S. Abrahams and J.M. Bacon. A software implementation of kimbrough’s disquotation theory for representing and enforcing electronic commerce contracts. *Group Decision and Negotiation*, 11(6):487 – 524, November 2002.
- [2] S. Chalmers, A. Preece, T. J. Norman, and P.M.D. Gray. Commitment management through constraint reification. In Nicholas R. Jennings, Carles Sierra, Liz Sonenberg, and Milind Tambe, editors, *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, volume 1, pages 430–437, Columbia University, New York, July 2004.
- [3] Aspasia Daskalopulu, Theo Dimitrakos, and Tom Maibaum. Evidence-based electronic contract performance monitoring. *Group decision and negotiation*, 11(6):469–485, November 2002.
- [4] V. Dignum, J. J. Meyer, F. Dignum, and H. Weigand. Formal specification of interaction in agent societies. In *Proceedings of the second Goddard workshop on formal approaches to agent based systems (FAABS)*, pages 37–52, October 2002.
- [5] T. Dimitrakos, I. Djordjevic, Z. Milosevic, A. Josang, and C. I. Phillips. Contract performance assesment for secure and dynamic virtual collaborations. In *Proceedings from the enterprise distributed object computing workshop (EDOC 2003)*, 2003.
- [6] A Josang. Subjective evidential reasoning. In *proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 2002)*, pages 1671–1678, Annecy, France, July 2002.
- [7] M. Kollingbaum and T. Norman. Supervised interaction – creating a web of trust for contracting agents in electronic environments. In *Proceedings of the First iInternational Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.
- [8] Peter McBurney and Simon Parsons. Chance discovery using dialectical argumentation. In *Proceedings of the Joint JSAI 2001 Workshop on New Frontiers in Artificial Intelligence*, pages 414–424. Springer-Verlag, 2001.
- [9] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, W. A. Gray, and N. J. Fiddian. Conoise: Agent-based formation of virtual organisations. In *Research and Development in Intelligent SystemsXX: Proceedings of AI2003, the Twentythird SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 353–366. Springer-Verlag, 2003.
- [10] H. Prakken and G. Sator. *Computational Logic: Logic Programming and Beyond. Essays In Honour of Robert A. Kowalski, Part II*, pages 342–380. Number 2048 in Lecture notes in Computer Science. Springer, Berlin, 2002.

- [11] Henry Prakken. Analysing reasoning about evidence with formal models of argumentation. *Law, Probability & Risk*, 3(1):33–50, 2004.
- [12] Henry Prakken, Chris Reed, and Doug Walton. Argumentation schemes and burden of proof. In *Workshop Notes of the ECAI-04 Workshop on Computational Models of Natural Argument*, Valencia, Spain, 2004.
- [13] C. A. Reed and G. W. A. Rowe. Araucaria: Software for argument analysis, diagramming and representation. *International Journal of AI Tools* 2, 14(3-4), 2005. to appear.
- [14] D. Reeves, B. Grosz, M. Wellman, and H. Chan. Toward a declarative language for negotiating executable contracts. In *Proceedings of the AAAI-99 Workshop on Artificial Intelligence in Electronic Commerce (AIEC-99)*, pages 39–45, 1999.