Modeling Conversation Policies using Permissions and Obligations

Lalana Kagal and Tim Finin

Computer Science and Electrical Engineering Department
University of Maryland Baltimore County
Baltimore, Maryland, USA
{lkagal1,finin}@cs.umbc.edu

Abstract. We describe our preliminary work in modeling conversation specifications and policies as positive/negative permissions and obligations. Our model is generic as it is independent of the syntax and semantics of the communication language and can be used for different agent communication languages. We also discuss the relationship between conversation specifications and policies and show how both are used by an agent in order to decide what communicative act to perform next within a conversation. Our work is different from existing research in communication policies because it is not tightly coupled to any domain information like the mental states of agents or specific communicative acts.The main contributions of this work include (i) an extensible framework that can support varied domain knowledge and different agent communication languages, and (ii) the declarative representation of conversation specifications and policies in terms of permitted and obligated speech acts.

1 Introduction

Multi-agent systems assume that agents interact and collaborate to satisfy their goals. Agent communication plays a very important part in these systems. A *conversation* can be defined as a sequence of communicative acts exchanged between interacting agents towards satisfying a particular goal [7, 9, 18]. In order for a conversation to be meaningful, it should follow some structured specifications. However, these conversation specifications or interaction protocols solely define the order in which communicative acts can be performed and do not take into consideration the content of the message, the attributes of the sender or the recipient or any other context. Similar to Phillips [18], we propose that along with conversation specifications, agents should use policies that define constraints over different aspects of the conversation in order to provide more flexible control over agent communication. This also allows the communication modules of agents to be less dependent on the communication protocols permitting the modification of conversation specifications and policies without requiring the modules to be changed.

We differentiate between conversation specifications that define the order of communicative/speech acts and policies that affect how conversation specifications are used and how conversations are carried out. *Conversation specifications*, or interaction protocols as they are known within FIPA [6], define the order in which communicative acts can occur within a conversation. For example, on receiving a REQUEST communicative act, an agent can reply with REFUSE or AGREE [6]. On the other hand, we define *conversation policies* as restrictions on the conversation based on the content of the communicative act, the attributes of the sender and recipient including their beliefs, desires and intentions and other context like the current team they belong to, the time of day, and their location. For example, a conversation policy would *oblige* an agent to provide an evasive answer to a QUERY about a political issue in an office setting but *permit* it to provide a more truthful answer in a social setting. However, we also consider other policies like privacy, work, and social that may establish additional restrictions and limitations on the communicative capabilities of the agent. Consider an agent that has a privacy policy prohibiting it from disclosing the SSN of the user. Though the conversation specification provides the set of communicative acts the agent can use to reply to a QUERY, its privacy policy prohibits it from responding to any query involving the SSN of the user.

In this paper, we describe our preliminary work in modeling conversation specifications and policies as positive/negative permissions and obligations. We also describe mechanisms for resolving conflicts between specifications and policies that enable an agent to decide what communicative act to perform next within a conversation. Our work is different from existing research in communication policies because it is not tied to any domain information like the mental state of the agent or to specific communicative acts [5, 9, 21, 22]. We try to provide an extensible framework that can be used to develop conversation protocols and policies over different kinds of domain-specific knowledge and different agent communication languages.

As an example, we describe the recent issue with the Medicare prescription drug bill in the United States [2] in terms of agent communication. According to the CNN article, Rick Foster, chief actuary for the Centers for Medicare and Medicaid Services, stated that he was asked not to answer questions from congressional Democrats regarding the cost of the bill before a series of key votes last summer. We describe how this would have worked within a multi-agent system driven by our conversation specifications and policies. Agents, including Foster, would have a conversation specification that states that in response to a QUERY, the agent is *permitted* to use either AGREE followed by an INFORM/FAILURE or REFUSE or ignore the message. The work policy would state that all state employees are *obliged* to answer queries from the congressional Democrats. However, agency chief Thomas Scully, enforces a temporary policy of the highest priority on Foster stating that Foster is *obliged* to REFUSE all queries from congressional Democrats regarding the estimated cost of the Medicare prescription drug bill until the end of summer. There also exists a sanction associated with the failure to fulfill this obligation which states that Foster could lose his job.

Whenever Foster receives a message, he reasons over his conversation specifications and policies to figure out how he should respond. When he receives a QUERY from a congressional Democrat asking about the estimated cost of the bill he knows from the conversation specifications that the correct response is AGREE or REFUSE. As his work policy *obliges* him to answer all queries from congressional Democrats, under normal circumstances Foster would agree. However, as Scully's temporary policy overrides the work policy and because of the associated sanction, Foster follows Scully's policy and REFUSEs the query. Scully's policy could also include rules *obliging* Foster to send an evasive reply to the congressional Democrats instead of refusing to answer.

2 Framework

A communicative or speech act is defined in terms of the set of actions that are implied when an agent makes an utterance. Generally, there are three actions that can be identified; (i) locution, which is the action of uttering the speech act, (ii) illocution, which deals with the conveying of the intentions of the sender, and (iii) perlocution, which are actions that occur due to the illocution. Conversation specifications define the sequence in which communicative acts can be performed in order for agents to have a meaningful dialogue. We model them as a set of permissions and obligations on speech acts based on the communicative acts exchanged thus far. We believe that conversation specifications should be very simple and only provide a list of possible speech acts that can be performed at a given time. This list of possible acts is then restricted by the policies acting on the agent. However, within our framework it is also possible to develop complex specifications that resemble policies.

Though our work has been done in OWL [23], a web ontology language used to describe metadata about entities, for conciseness and ease of explanation, we use expressions in predicate logic to describe speech acts, positive and negative deontic objects, and policies.

- A *communicative or speech act* is performed by an agent to achieve a certain intention. A speech act is usually assumed to have two main components; the performative and the proposition.
  We describe a communicative act as a tuple

  ```
  performative(Sender, Receiver, Proposition)
  ```

  For example, a QUERY-REF speech act of FIPA sent from agentX to agentY asking agentY what he believes the values of the included proposition to be

  ```
  query-ref(agentX, agentY, estimatedCostOfBill(Cost))
  ```

- Domain actions are actions that an agent can perform and are described by the following tuple

  ```
  action(Actor, Target, PreCondition, Effect)
  ```

The *printAPage* domain action can be described as

```
printAPage(X, hpLaitPrinter,
     (numPages(hpLaitPrinter, N), N>0), (numPages(hpLaitPrinter, N-1)))
```

– Deontic concepts of permissions, prohibitions (negative permissions), obligations and dispensations (waiver from an obligation) are used to describe the behavior of the agent.

```
deontic(Actor, Action, Constraint)
or
deontic(Actor, Action, StartingConstraint, EndingConstraint)
```

Consider the permission of an agent to perform an AGREE speech act to any agent regarding for the estimated cost of the Medicare prescription bill. This is considered a policy as it includes domain knowledge of the proposition used to model the cost of the bill.

```
permission(X, agree(X, Y, estimatedCostOfBill(Cost)), _)
```

We model 4 deontic objects: permission, prohibition, obligation and dispensation. Permissions and prohibitions are used to describe positive/negative authorizations whereas obligations and dispensations describe positive/negative responsibilities. All these objects could be represented in terms of a single concept, either permission or obligation, but we use different terms for simplicity.
Associated with each deontic object is either *constraint*, which defines the conditions under which the deontic object is applicable, or *startingConstraint* and *endingConstraint* that define the window within which the deontic object is applicable. These constraints could also include conditions on time providing time validity to the deontic object. Obligations and dispensations have an additional field, obligedTo, which describes whom the agent is obliged to. Another property called sanctions is associated with both obligations and prohibitions and is used to describe the penalties imposed on the agent if it fails to fulfill the obligation or violates the prohibition. Consider a policy of a graduate assistant that obliges him to turn in a weekly status report to his advisor or risk missing a pay check.

```
obligation(X, inform(X, Y, weeklyStatus(X, W, Status)),
          (advisor(Y, X), endOfWeek(W)), Y, missPayCheck(X, W))
```

A *permission* allows an agent to perform the associated action as long as the constraint is true or the startingConstraint is true and the endingConstraint is false. A *prohibition* prevents an agent from performing the associated action as long as the constraint is true or during the time when the startingConstraint is true and the endingConstraint is false. An agent must perform an *obligation* sometime before the *endingConstraint* is false and after the *startingConstraint* is true. An agent is no longer obliged to fulfill an obligation if there is an associated dispensation freeing the agent from the obligation.

– In our framework conflicts can occur between permissions and prohibitions, obligations and prohibitions, and obligations and dispensations. In order to resolve conflicts, our framework includes meta-policies that are used to correctly interpret policies. There are two kinds of meta-policies namely setting the modality precedence (negative over positive or vice versa) or stating the priority between rules within a policy or between policies [17].
In a multi-policy environment, it is possible to state that one policy overrides another. For example, it is possible to say that in case of conflict the CS department policy always overrides the Lait lab policy. As another example, consider the CS department policy. Students are prohibited from using the faculty printer but research assistants are permitted to. There is a potential conflict if a student is a research assistant and needs to use the faculty printer. This can be solved by setting the priority between the rules and stating that the permission overrides the prohibition.

```
rule1 : prohibition(X, print(X, facultyPrinter), student(X))
rule2 : permission(X, print(X, facultyPrinter), researchAssistant(X))
overrides(rule2, rule1)
```

On the other hand, if a certain modality precedence is used, then when a conflict occurs the rule with the preferred modality overrides the other. For example, if positive modality is preferred then in case of conflict, permissions and obligations will override prohibitions and obligations will override dispensations. The conflict in the CS department policy in the earlier example can also be resolved if positive modality is given precedence.

```
precedence(positive-modality)
```

– We also use some additional expressions to describe the sequence of message that have been exchanged so far in an actual dialogue. The expression

```
received(X)
```

states that X was a message received and

```
sent(X)
```

states that X was a message that was sent.

3 Conversation Specifications

Using the semantics of the deontic objects and domain actions and the syntax of speech acts, we can model conversation specifications in agent communication languages like Knowledge Query and Manipulation Language (KQML) [5, 13] or Foundation for Intelligent Physical Agents (FIPA) [6] as a set of permissions and obligations on the sender or the receiver depending on the performatives used thus far in the conversation.

As an example, we describe the QUERY-REF specification in FIPA.

- Speech acts used : QUERY-REF, REFUSE, AGREE, FAILURE, INFORM
- Sequence of messages : An agent sends a QUERY-REF message to another agent. The latter can reply either with a REFUSE or an AGREE stating its intent to either provide an answer or refuse to answer. Once an agent has sent an AGREE, it is obliged to send an INFORM providing the information required.
  - Every agent has the permission to perform a QUERY-REF performative
    `permission(X, query-ref(X, Y, Proposition),_)`
    In the above expression, the constraint field is left empty to specify that there are no constraints on the performing of a QUERY-REF performative.
  - On receiving a QUERY-REF, the recipient has the permission to either REFUSE the query or AGREE to provide the answer
    ```
    permission(Y, refuse(Y, X, Proposition),
              received(query-ref(X,Y, Proposition)))
    permission(Y, agree(Y, X, Proposition),
              received(query-ref(X,Y, Proposition)))
    ```
    The constraint here is that the agent has received a QUERY-REF speech act.
  - Once an agent has accepted a QUERY-REF, it is obliged to answer to it either with a FAILURE or with an INFORM and the agent is obligated to the recipient of the agree message.
    ```
    obligation(Y, failure(Y, X, Proposition),
              sent(agree(Y, X, Proposition)), X, _)
    obligation(Y, inform(Y, X, Proposition),
              sent(agree(Y, X, Proposition)), X, _)
    ```

Other specifications are simpler like the FIPA PROPOSE interaction protocol.

- Speech acts used : PROPOSE, REJECT-PROPOSAL, ACCEPT-PROPOSAL
- Sequence of messages : An agent sends a PROPOSAL message to another agent. The recipient can either use the REJECT-PROPOSAL or the ACCEPT-PROPOSAL.
  - Every agent has the permission to perform a PROPOSAL performative
    `permission(X, proposal(X, Y, Proposition), _)`
  - On receiving a PROPOSAL, the recipient has the permission to either reject the proposal or accept it.
    ```
    permission(Y, accept-proposal(Y, X, Proposition),
              received(proposal(X,Y, Proposition)))
    permission(Y, reject-proposal(Y, X, Proposition),
              received(proposal(X,Y, Proposition)))
    ```
    The constraint here is that the agent has received a proposal speech act.

4 Policies

Policies like conversation, social, and privacy add restrictions on the performatives that can be used, the content of the speech act, the receiver, time of the message, etc. based on current attributes of the sender, receiver, content and all other context of the conversation. Policies can be defined at two levels; one that is independent of the syntax and semantics of the communication language and the second that is tightly integrated with them. In the latter case, the policies use the semantics of the performative and define constraints on how performatives can be used and under what conditions. Though this may be true in the case of conversation policies, we generally assume that policies like privacy, and social norms define restrictions at the higher level of abstraction and provide restrictions on the general behavior of the agent. Whenever these policies deal with information flow between agents, they need to be translated into lower level policies using the semantics of the communication language. For example, an agent's privacy policy might state that the SSN must not be disclosed. This is irrespective of the agent communication language being used or the specific performative. If FIPA is being used, the privacy policy could be translated in our framework as 'The agent is prohibited from sending an INFORM communicative act to any agent when the content involves SSN of the agent'. However, if KQML is the language being used for communication, the semantics specify that only the TELL is the only assertive performative that causes the agent to reveal its belief about a proposition. In this case, the policy could translate to 'The agent is prohibited from sending a TELL communicative act to any agent when the content involves SSN of the agent'. Similarly, a social policy can specify that an agent should not be rude. However, what it means to be rude and how it translates into speech acts and their content depends on the application domain. The agent would have to ensure that the effect of any speech act does not violate this social policy.

Following from the first example dealing with the Medicare bill, it is evident that there are several policies acting on an agent. This could lead to *conflicts* between policies. Foster's conversation specifications gave him the permission to reply to requests, however, the agency head prohibited him from replying to queries about the estimated cost of the Medicare bill. In Foster's case, Scully's policy would be enforced if it was of higher priority than Foster's other policies.

5 Example

We now walk through the Medicare bill example. We assume that the agent communication language used is FIPA and both Foster and Scully share the same conversation specifications. These specifications include the QUERY-REF specification described in section 3.

- Foster has a work conversation policy that specifies that he should respond to all queries from congressional Democrats.

```
ConvPolicy :
obligation(X, agree(X, Y, Proposition),
        (received(query-if(Y, X, Proposition)), congressionalDemocrat(Y)),
        X, _)
obligation(X, agree(X, Y, Proposition),
        (received(query-ref(Y, X, Proposition)), congressionalDemocrat(Y)),
        X, _)
```

– Scully decides that Foster should not answer any queries from congressional Democrats that ask about the estimated cost of the Medicare prescription bill. This is a high level policy and could be translated in two ways; either as an obligation to use REFUSE or a prohibition on INFORM.

1. It can be translated based on the syntax and semantics the FIPA ACL as an obligation to refuse all queries about the estimated cost of the bill from congressional Democrats.

```
TempPolicy :
obligation(X, refuse(X, Y, estimatedCostOfBill(Cost)),
        (received(query-ref(Y, X, estimatedCostOfBill(Cost))),
          congressionalDemocrat(Y)),
         scully, loseJob(foster))
```

2. It can also be translated as a prohibition from informing any congressional Democrat about the estimated cost of the bill.

```
TempPolicy :
prohibition(X, inform(X, Y, estimatedCostOfBill(Cost)),
        (received(query-ref(Y, X, estimatedCostOfBill(Cost))),
         congressionalDemocrat(Y)))
```

However, we believe that the former interpretation matches the statement made by Foster more closely, so we use it through the rest of the example.

– Scully gives this obligation policy higher priority than the existing conversation policy.

```
overrides(TempPolicy, ConvPolicy)
```

– At some point of time, a congressional Democrat, Walter, sends a query to Foster asking about the estimated cost of Medicare bill.

```
query-ref(walter, foster, estimatedCostOfBill(Cost))
```

– On receiving this speech act, Foster looks up the conversation specifications for QUERY-REF, and finds that he can respond either with an AGREE or REFUSE.
– Foster checks his work conversation policy, which states that he is obliged to answer all QUERY-IF and QUERY-REF speech acts from congressional Democrats with an AGREE.

– Foster then reasons over Scully's policy that is of higher priority than his work conversation policy. Scully's policy states that Foster is obliged to refuse all queries from congressional Democrats about the estimated cost of the bill. As Scully's policy is of higher priority and as the cost of violating the policy involves Foster losing his job, Foster uses Scully's policy and sends a REFUSE to Walter.

```
refuse(foster, walter, estimatedCostOfBill(Cost))
```

6 Specification Language

Our policy language, Rei, is represented in OWL [23], which is a ontology language. It includes logic-like variables to describe constraints over different aspects of deontic objects, actions and policies. The use of variables allows Rei to represent a wider range of constraints than would be possible in OWL. We used logic to describe the examples in this paper for ease of explanation and for conciseness. By using OWL, Rei gains extensibility as different kinds of domain specific knowledge in RDF and OWL can be used for describing policies and specifications.

Our policy language is modeled on deontic concepts of permissions, prohibitions, obligations and dispensations [11, 10]. We believe that most policies can be expressed as what an entity can/cannot do and what it should/should not do in terms of actions, services, and conversations, making our language capable of describing a large variety of policies ranging from security policies to conversation and behavior policies. The policy language has some domain independent ontologies but will also require specific domain ontologies. The former includes concepts for permissions, obligations, actions, speech acts, etc. The latter is a set of ontologies, used by the entities in the system, which defines domain classes (person, file, deleteAFile, readBook) etc. and properties associated with the classes (age, num-pages, email).

The language includes two constructs for specifying meta-policies that are invoked to resolve conflicts; setting the modality precedence (negative over positive or vice versa) or stating the priority between policies [15, 14]. As an example of using priority consider a meta policy that states that in case of conflict the Federal policy always overrides the State policy. When modality precedences are used to resolve a conflict, the rule of the preferred modality overrides the other.

Another important aspect of the language is that it models speech acts like delegation, revocation, request and cancel for modifying existing policies dynamically. Delegations and revocations cause the permissions of agents to be modified, whereas requests and cancels affect the obligations. A delegation speech act, if valid, causes a permission to be created. A revocation speech act nullifies an existing permission (whether policy based or delegation based) of an agent. An agent can request another agent for a permission or to perform an action on its behalf. The former if accepted causes a delegation and the latter leads to an obligation. An agent can also cancel any previously made request causing a dispensation.

7 Policy Enforcement

Along with the specification of Rei, we have also developed a reasoning engine in Flora [1], which is an F-logic extension of XSB. The engine is built over F-OWL [24], a reasoner for OWL and RDF, enabling Rei to understand and reason over policies and specifications in both OWL and RDF. The engine reasons over policies, meta policies, history of speech acts, and domain information to answer the following types of questions :

- What are the current permissions of X ?
  The engine looks for all those permissions whose actor property unifies with X and whose constraints are satisfied. If there is a conflicting prohibition or revocation, the engine uses the meta policies to decide whether the permission overrides the prohibition/revocation or vice versa. If the latter case is true, the permission is not valid. If the permission is valid, the policy engine checks the preconditions associated with the action over which the permission is specified. The permission is returned only if the precondition is satisfied.
  The engine also looks for valid delegations from any agent to X. The delegation is valid if the delegatee has the permission to make the delegation or has been delegated the permission to make the delegation. The entire delegation chain is checked by policy engine. At every level, the engine also checks that there is no conflicting prohibition or revocation.
- What are the current obligations of Y ?
  The engine locates all obligations whose actor property unifies with Y and whose startingConstraint is satisfied but whose endingConstraint is false. The engine ensures that there is no conflicting dispensation.
- Does X have the permission to perform action A or speech act S ?
  This is similar to the first case, but in this case, the policy engine also checks the action property of the permission and verifies that it unifies with A or S.
- Does X have any permissions on a resource R ?
  This is similar to the first case, but the policy engine also tries to unify the target property of the action associated with the permission with R.
- When policy P is deleted, does agent X still retain the permission to use the QUERY speech act for Proposition P ?
  This is part of the policy analysis provided by Rei. The policy engine deletes P but stores it in a temporary list. It then tries to verify that X has the permission to use QUERY speech act over P. It returns the answer and then restores P.

We envision that the reasoning engine will be used together with domain knowledge like the mental state of the agents, the history of the speech acts performed and other context by either a planning component or a workflow component to enable enforcement of policies over agent communication.

[1] http://flora.sourceforge.net

Using the policy engine, our earlier example of the Medicare bill would be inferred by Foster as

1. Received QUERY-REF from congressional Democrat enquiring about the cost of the Medicare bill
2. What are my current obligations ? I am obliged to AGREE to answer by my conversation work policy. I am also obliged to REFUSE the query that deal with the cost of the bill by Scully's policy.
3. As the meta policy states that Scully's policy has the highest priority, I execute it first.
4. So, I REFUSE the query.
5. However, Do I have the permission to REFUSE a query ? Yes, from the conversation specifications.

8 Related Work

Cohen and Levesque model the cognitive state of agents and base allowable speech acts on the cognitive states of collaborating agents [3]. In his earlier work, Singh provides semantics for speech acts in terms of beliefs and intentions of the agents [19, 20]. Fornara and Colombetti [8] describe an approach based on the notion of social commitment. Labrou and Finin also describe the semantics of KQML based on the beliefs and desires of agents [5, 13]. These models are very tightly coupled to the mental states of agents and the semantics of the language that makes it difficult to extend them to work in different environments and with different agent communication languages. Cost et al. [4] develop a model using colored petri nets that can take into account various contextual properties and attributes. Greaves et al. define conversation policies as restrictions on how the agent communication language is used [9]. Though the last approach is similar to ours, we believe that conversation policies should be at a higher level of abstraction and should not involve specifics of the communication language. We also propose that all policies related to communication be translated into permissions and obligations that the agent has on speech acts, which are supported by the communication language being used.

Kollingbaum et al. discuss how normative agents estimate the effect of adopting a new norm[12]. The current beliefs, norms and the selected plan are taken into consideration while estimating the level of consistency that will be brought about by the adopted norm. This work approaches the adoption of norms (or what we call policies) under the assumption that the agent can decide whether or not to accept a norm. Though this is advisable for contracting agents, we believe that certain policies are enforced by the environment and must be accepted by the agent irrespective of whether they cause conflicts or inconsistencies in the agent's current state. Also, Kollingbaum's approach does not try to resolve conflicts, it only categorizes the type of conflict in terms of consistency and uses this information to decide whether or not to accept a new norm.

Broersen et al. use agent types to resolve conflicts between beliefs, obligations, intentions and desires [1]. The agent types are determined by their characteristics

namely social (obligations overrule desires), selfish (desires overrule obligations), realistic (beliefs overrule everything else) and simple-minded (intentions overrule obligations and desires). In our framework, conflicts basically occur between permissions and prohibitions, obligations and prohibitions, and obligations and dispensations. In order to resolve conflicts, our framework includes meta-policies namely setting the modality precedence (negative over positive or vice versa) or stating the priority between rules within a policy or between policies. Broersen et al. approach conflict resolution from the agent's point of view whereas we try to resolve conflicts in policies within the environment and not within agents themselves. We believe that Broersen's approach or something similar could be used by agents after conflict resolution is provided by our framework as the enforced policies may conflict with the agent's internal beliefs, desires, intentions, obligations, prohibitions, and permissions.

9 Summary

In this paper, we do not try to define the semantics of a particular agent communication language or a set of performatives but provide a flexible model that can be used to describe conversation specifications and policies over different agent communication languages like KQML and FIPA and different domain-specific information. The framework allows specifications to be described as a sequence of permitted and obligated speech acts. Policies are described at a high level of abstraction and are translated into positive/negative permissions and obligations over speech acts using the semantics of the agent communication language. These permissions and obligations establish restrictions over attributes of the sender, receiver, content and other context of the conversation like time, and location. As part of our future work, we are looking into automating the translation process from high level policies to performative specific permissions and obligations.

Though we described all our examples in logic, our actual specification language is in OWL. We have developed a reasoning engine for our language that reasons over domain knowledge, speech act semantics, protocols, policies, and meta policies to answer questions about the permissions and obligations of an agent with respect to the actions and speech acts it can/should perform. We envision that this reasoning engine will be coupled with the planning/workflow component of an agent to provide policy enforcement over agent communication. We are also interested in integrating into our framework work on commitments like that by Mallya et al. [16], which involves reasoning over the status of obligations of agents.

References

1. Jan Broersen, Mehdi Dastani, Joris Hulstijn, Zisheng Huang, and Leendert van der Torre. The boid architecture conflicts between beliefs, obligations, intentions and desire. 2001.

2. Cable News Network (CNN). Probe under way on medicare cost. http://www.cnn.com/2004/ALLPOLITICS/ 03/17/medicare.investigation/, 2004.
3. Philip R. Cohen and Hector J. Levesque. Intention is choice with commitment. In *Artificial Intelligence*, 1990.
4. R. Scott Cost, Ye Chen, Tim Finin, Yannis Labrou, and Yun Peng. Using colored petri nets for conversation modeling. In *Agent Communication Languages, Frank Dignum and Mark Greaves (editors), Springer-Verlag, Lecture Notes in AI, 2000*, 2000.
5. Tim Finin, Richard Fritzson, Don McKay, and Robin McEntire. A semantics approach for kqml – a general purpose communication language for software agents. In *Third International Conference on Information and Knowledge Management (CIKM'94)*, November 1994.
6. FIPA. Foundation for intelligent physical agents specifications. http://www.fipa.org.
7. R.A. Flores and R.C Kremer. A model for flexible composition of conversations: How a simple conversation got so complicated. In *3rd Workshop on Agent Communication Languages and Conversation Policies, M.P. Huget, F. Dignum and J.L. Koning (Eds.), First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2002), Bologna, Italy, July 15-19, 2002*, July 2002.
8. Nicoletta Fornara and Marco Colombetti. Defining interaction protocols using a commitment-based agent communication language. In *Second international joint conference on Autonomous agents and multiagent systems, Melbourne, Australia pp 520-527, 2003, ACM Press*, 2003.
9. Mark Greaves, Heather Holmback, and Jeffrey Bradshaw. What is a conversation policy? In *Autonomous Agents '99 Workshop on Specifying and Implementing Conversation Policies*, 1999.
10. Lalana Kagal, Tim Finin, and Anupam Joshi. A policy based approach to security for the semantic web. In *2nd International Semantic Web Conference (ISWC2003), September 2003*, 2003.
11. Lalana Kagal, Tim Finin, and Anupam Joshi. A policy language for pervasive systems. In *Fourth IEEE International Workshop on Policies for Distributed Systems and Networks*, 2003.
12. M. J. Kollingbaum and T.J. Norman. Norm consistency in practical reasoning agents. 2003.
13. Yannis Labrou and Tim Finin. A semantics approach for kqml – a general purpose communication language for software agents. In *Third International Conference on Information and Knowledge Management (CIKM'94)*, November 1994.
14. Emil C. Lupu and Morris Sloman. Towards a role based framework for distributed systems management. *Journal of Networks and Systemss Management, Plenum Press*, 1996.
15. Emil C. Lupu and Morris Sloman. Conflicts in policy-based distributed systems management. *IEEE Transactions on Software Engineering*, 1999.
16. Ashok U. Mallya, Pinar Yolum, and Munindar P. Singh. Resolving commitments among autonomous agents. In *International Workshop on Agent Communication Languages and Conversation Policies (ACL), Melbourne, July 2003, Springer*, 2003.
17. Jonathan Moffett and Morris Sloman. Policy conflict analysis in distributed systems management. *Journal of Organizational Computing*, 1993.
18. Laurence R. Phillips and Hamilton E. Link. The role of conversation policy in carrying out agent conversations. In Frank Dignum and Mark Greaves, editors, *Is-*

sues in Agent Communication, volume 1916 of *Lecture Notes in Computer Science*. Springer, 2000.
19. M.P. Singh. Towards a formal theory of communication for multiagent systems. In *IJCAI'91*, 1991.
20. M.P. Singh. A semantics for speech acts. *Annals of Mathematics and Artificial Intelligence*, 1992.
21. Ira A. Smith and Philip R. Cohen. Toward a semantics for an agent communication language based on speech acts. In Howard Shrobe and Ted Senator, editors, *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference, Vol. 2*, pages 24–31, Menlo Park, California, 1996. AAAI Press.
22. Ira A. Smith, Phillip R. Cohen, Jeffery M. Bradshaw, Mark Greaves, and Heather Holmback. Designing conversation policies using joint intention theory. In *Third International Conference on Multi-Agent Systems (ICMAS98)*, 1998.
23. W3C. Owl web ontology language. http://www.w3.org/2001/sw/WebOnt/, 2004.
24. Youyong Zou, Harry Chan, and Tim Finin. F-owl: an inference engine for semantic web. In *Third NASA-Goddard/IEEE Workshop on Formal Approaches to Agent-Based Systems (FAABS III), 26-28 April 2004, Greenbelt MD*.