Ontology Negotiation between Agents Supporting Intelligent Information Management

Sidney C. Bailin
Knowledge Evolution, Inc.
1050 17th Street, NW, Suite 520
Washington DC 20009
sbailin@kevol.com

Walt Truszkowski
NASA/Goddard Space Flight Center
Goddard Space Flight Center Code 588
Greenbelt, MD 20771
Walt.Truszkowski@gsfc.nasa.gov

1 ABSTRACT

This paper describes an approach to *ontology negotiation* between agents supporting intelligent information management. Ontologies are declarative (data driven) expressions of an agent's "world": the objects, operations, facts, and rules that constitute the logical space within which an agent performs. Ontology *negotiation* enables agents to cooperate in performing a task, even if they are based on different ontologies.

Our objective is to increase the opportunities for "strange agents" — that is, agents not necessarily developed within the same framework or with the same contextual operating assumptions — to communicate in solving tasks when they encounter each other on the web. In particular, we have focused on information search tasks.

We have developed a protocol that allows agents to discover ontology conflicts and then, though incremental interpretation, clarification, and explanation, establish a common basis for communicating with each other. We have implemented this protocol in a component that can be added to a variety of agents, irrespective of their underlying ontological assumptions.

This paper will present an overview of the Ontology Negotiation Protocol (ONP) and a brief discussion of its implementation in Jess (Java Expert System Shell). Much remains to be done. The protocol must be tested in larger and less familiar contexts (for example, numerous archives that have not been pre-selected) and it must be extended to accommodate additional forms of clarification and ontology evolution.

2 GOALS AND BENEFITS

Ontology negotiation is a new concept; there is very little written on the subject. However, it is becoming increasingly recognized as a crucial element of scalable agent technology. This is because agents, by their very nature, are supposed to operate with a fair amount of autonomy and independence from their end-users. Part of this independence is the ability to enlist other agents for help in performing a task (such as locating information on the web). The agents enlisted for help may be "owned" by a different end-user or organization (such as a document archive), and there is no guarantee that they will use the same terminology or understand the same concepts (objects, operators, theorems, rules) as the recruiting agent.

For NASA, the need for ontology negotiation arises at the boundaries between scientific disciplines. For example: modeling the effects of global warming might involve knowledge about imaging, climate analysis, ecology, demographics, industrial economics, and biology. The need for ontology negotiation also arises at the boundaries between scientific programs. For example, a Principal Investigator may want to use information from a previous mission to complement downloads from the instruments currently deployed.

The need for ontology negotiation is being recognized in commercial circles as well, especially e-business. Here it arises from the rapid rate at which new capabilities (agents) are appearing, and the consequent unfeasibility of establishing and enforcing a standard ontology. The need is especially strong in cross-industry transactions. Even if standard ontologies are developed for particular industries, solutions that involve multiple vertically or horizontally integrated domains will still require a more dynamic way to resolve ontological mismatches.

We have taken a first step towards enabling agents to resolve ontology conflicts dynamically. Specifically, we have tested the ONP with queries against two scientific databases: NASA's Global Change Master Directory and the National Oceanic and Atmospheric Administration (NOAA) Wind and Sea index. The tests demonstrate how the ONP allows "strange" agents to coordinate a response to an information request.

The emphasis on protocol distinguishes our work from other research into ontology conflict. While efforts

such as (Clark and Porter, 1997; Cybenko and Jiang, 1999; Heflin *et al*, 1999; Takeda *et al*, 1996) focus on defining or automating corrective actions, our goal is to automate the *dialogue* leading to such actions.

3 OVERVIEW OF THE PROTOCOL

Our goal was to specify and implement a robust method of ontology negotiation that allows web-based information agents to resolve semantic mismatches in real time without human intervention. The resulting software provides "strange" agents with a means of arriving at a common language in which to converse.

"Common language" refers not just to syntax, but also to the *meaning* of terms exchanged by the agents. The meaning of a term can be represented by an ontology in any of several forms: e.g., as facts pertaining to the term, as rules governing the use of the term, or as structural information relating the term to other terms (e.g., the "is-a" relation).

Between information agents, the "terms exchanged" consist primarily of the *query content* and *document descriptors* for the query results. Both of these can be viewed as keywords describing the document that is either desired (query) or found (results). Our research to date indicates that there are three processes involved in this type of negotiation:

- *Clarifying* the meaning of keywords
- *Explaining* the relevance of the query results
- *Evolving* an agent's ontology on the basis of clarification and relevance explanation

In addition, agents need to be able to locate other agents capable of performing specific search, fusion, or filtering tasks. Other researchers have extensively explored the mechanisms for discovering an agent with specific capabilities and engaging it in a task. Their research, however, has tended to ignore the problems of semantic mismatch: When an agent announces that it has (or needs) a particular capability, will other agents necessarily understand the terms in which the capability is described? The question suggests that semantic clarification, and therefore ontology negotiation, should play a role in the negotiation of capabilities and assignments, as much as in the satisfaction of information requests.

3.1 Objects of the Protocol

The ONP is based on a small set of object types and operations on those types. The available object types are presented in Table 1; some of them warrant explanation. *Queries* are requests to locate documents and/or URLs that are relevant to a set of descriptors (keywords). *Declination* expresses an agent's unwillingness to respond to a query (for reasons of capacity or capability), while *rejection* is an agent's expression of dissatisfaction with the current results of a query. These objects are all message types, whose relationships are illustrated in Figure 1.

Confirmation of interpretation is an agent A's validation of an agent B's tentative understanding of a previous message from A. *Clarification* is the means by which an agent makes explicit the meaning of a previous message it sent. These objects are also message types, whose relationships are illustrated in Figure 2.

An agent receives the results of a query as a set of URLs and document descriptors. It uses the descriptors (keywords) to *evaluate* the relevance of the results to the query. If it cannot see why a particular URL is relevant to the query, it can ask the agent that returned the results for an *explanation of relevance*. The structure of such an explanation is discussed in Section 3.3; here we just note that the explanation may include facts that are true in the server agent's ontology, and these may also require explanation (Figure 3).

3.2 Contextual Assumptions

The message types are defined as specializations of KQML performatives (Finin *el al*, 1997). The protocol establishes additional semantics associated with the specializations. For example, the various request types are instances of the KQML `ask` performative, where the KQML `content` consists of an *operation* — one of `ReqCla`, `ReqCfi`, `ReqCap`, `ReqExr`, or `ReqExf` (see Table1) — and an *interpretable*, which is the `Query`, `Query Result`, or `Capability Statement` for which a clarification, interpretation or explanation is sought.

The ONP could easily be adapted to work over the Agent Communication Language of the Foundation for Intelligent Physical Agents (FIPA, 1998). As a particular use of these constructs, the ONP represents a commitment to a certain style of communication. The commitment does not, however, interfere with an agent's ontological assumptions or with its patterns of communication outside the realm of ontology negotiation.

The objects and operations, while constituting a kind of ontology in their own right, do not usurp the role of an agent's domain ontology. They represent, in effect, an ontology for ontology negotiation. This level of commitment to some basic communication primitives specifically for the purpose of ontology negotiation is, we believe, essential, and does not lessen the benefits provided by the ONP in enabling agents to coordinate cross-domain problem-solving activities.

Table 1. Objects of the Ontology Negotiation Protocol.

Object Type	Abbreviation	Object Type	Abbreviation
Query	Que	Confirmation of Interpretation	Cfi
Query Results	Qre	Clarification	Cla
Acknowledgement	Ack	Explanation of Relevance	Exr
Rejection	Rej	Explanation of Fact	Exf
Declination	Dcl	Ontology	Ont
Capabilities Statement	Cap		

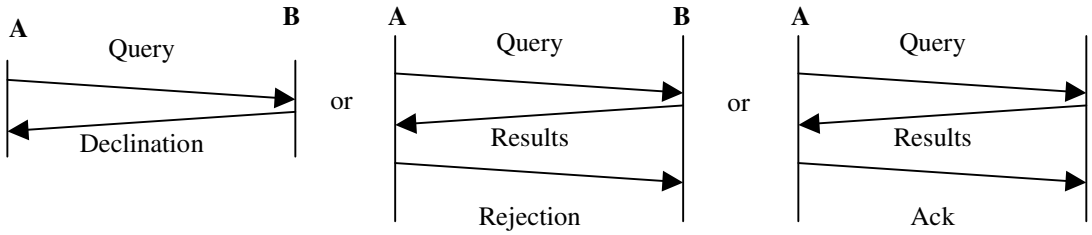

Figure 1. High-level structure of a query process. A is a human user or agent. B is an agent. Time flow is from top to bottom.

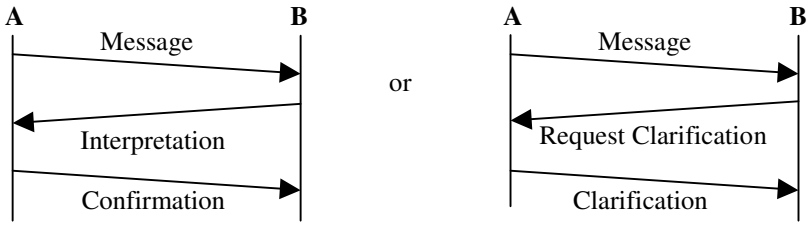

Figure 2. Interpretation and clarification in the ONP. Initial "message" can be query, query results, request for capability statement, or capability statement. "Interpretation" is actually a request for confirmation of interpretation in terms of Table 1.

.

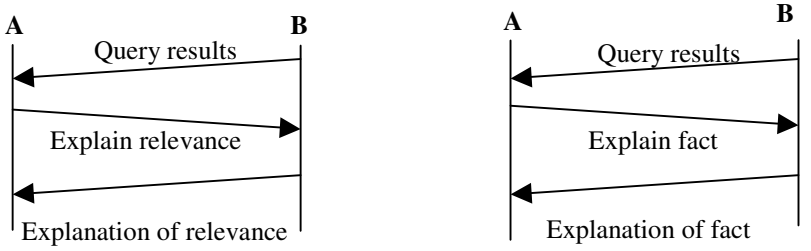

Figure 3. Evaluation and justification of query results.

3.3 States and Transitions

States of the ONP correspond to performance of one or another operation on a particular type of object. The available operations are shown in Table 2. Obviously not every combination of an operation and object represents a meaningful state. Table 3 presents the states of the ONP, using the abbreviations listed in Tables 1 and 2.

For example, an agent can wait for a capability statement (if it has requested one) but it cannot wait for a declination. Similarly, an agent can receive a declination from another agent, but it cannot receive an ontology

The behavior of agents participating in the protocol is determined by transitions between the possible states. Since there are 46 states defined in the ONP, the transitions are selected from the 46 x 46-element matrix of state pairs. Figures 4 through 6 present the transitions in several chunks (submachines). The triggering conditions are omitted for clarity.

The range of possible transitions presents a challenge in implementing the protocol—especially to implement it in a maintainable fashion. We have addressed this problem through a tool that generates the state machine code from a high-level specification.

Table 2. Operations of the Ontology Negotiation Protocol.

Operation	Abbreviation	Operation	Abbreviation
Wait for	Wtg	Return	Ret
Request	Req	Interpret	Int
Receive Request for	Rrf	Interpret Request for	Irf
Receive	Rcd	Evaluate	Evl
Process	Pro	Evolve	Evo
Forward	Fwd		

Table 3. States of the ontology negotiation protocol.

	Que	Qre	Ack	Rej	Dcl	Cap	Cfi	Cla	Exr	Exf	Ont
Wtg	x	x	x			x	x	x	x	x	
Req						x	x	x	x	x	
Rrf						x	x	x	x	x	
Rcd	x	x	x	x	x	x	x	x	x	x	
Pro	x										
Fwd	x										
Ret		x	x	x	x	x	x	x	x	x	
Int	x	x				x					
Irf						x					
Evl		x				x					
Evo											x

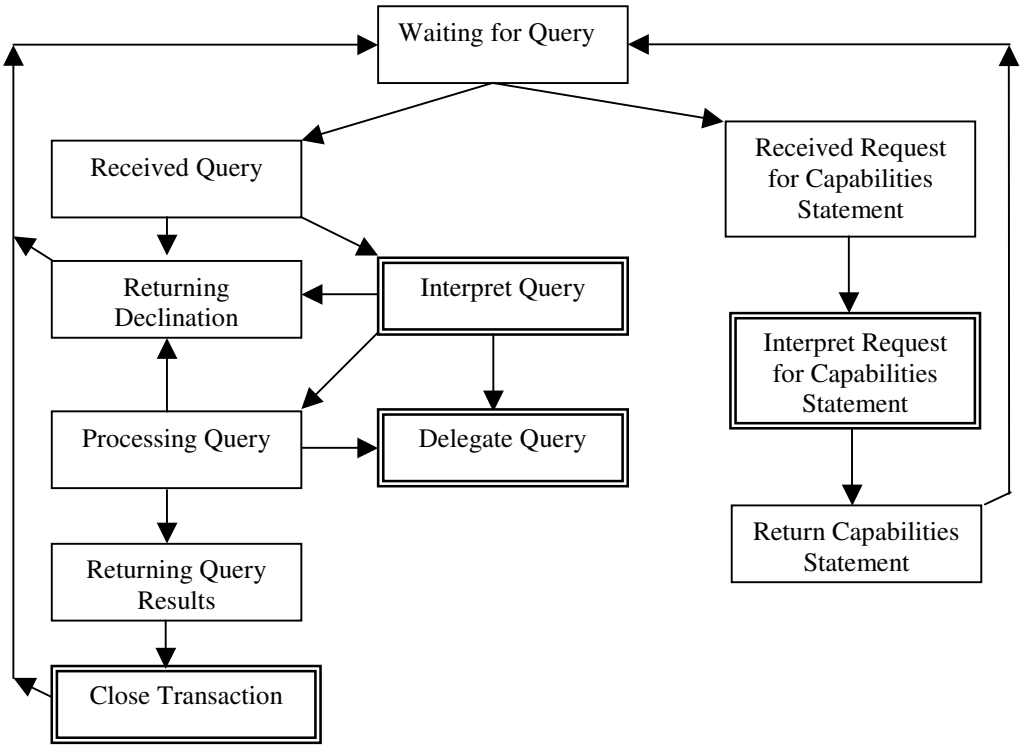

Figure 4. State transitions for agent acting as server. The double-bordered boxes represent submachines

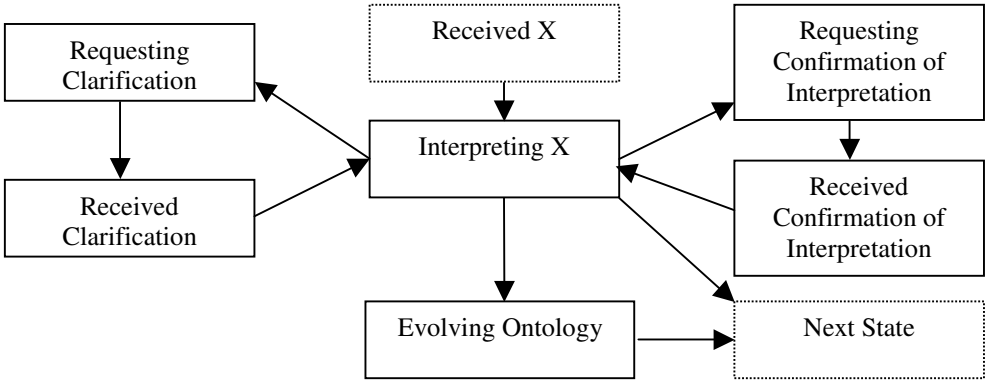

Figure 5. State transitions for message interpretation.

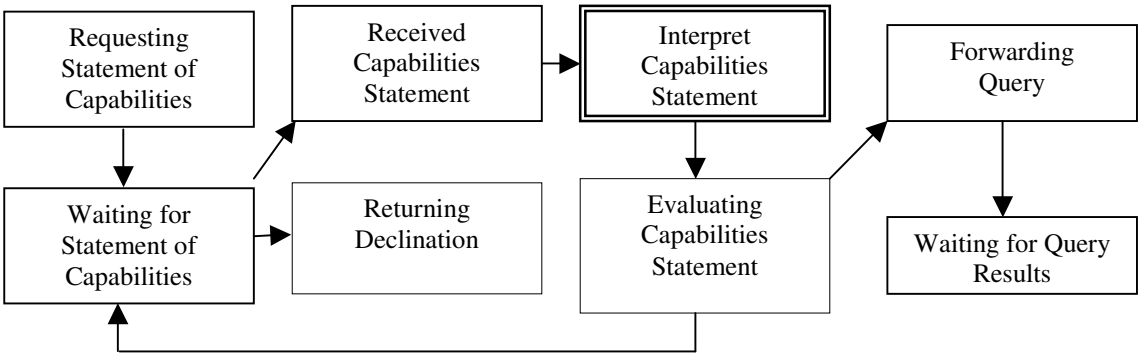

Figure 6. State transitions for delegating a query.

4 THE ONTOLOGY NEGOTIATION PROCESS

The state machine structure determines the shape of ontology negotiations. It does not determine how an agent decides whether to ask for clarification, or how an agent chooses to clarify a previous message. This is the heart of the ontology negotiation process. The fundamental tasks are interpretation, clarification, relevance evaluation, and ontology evolution.

4.1 Interpretation

Interpretation is the process of determining whether a message just received is properly understood. In the testbed, messages are sequences of keywords, and the recipient agent tries to interpret each keyword in turn. First, the agent checks its own ontology to see whether the keyword occurs there. If not, the agent queries the WordNet lexical database to find synonyms of the keyword (Fellbaum, 1998). Then it checks the ontology for any of these synonyms.

If a synonym is located in the ontology, it represents an interpretation of the keyword. Since WordNet may identify distinct meanings for the keyword (homonyms), each synonym is only a possible interpretation, which must be confirmed by the source of the message. The recipient agent therefore requests a *confirmation of interpretation* from the source agent.

4.2 Clarification

If an agent is not able to interpret some of the keywords of a message it has received, it can decide to proceed anyway (if enough other keywords are understood), or it can request a *clarification* from the source of the message. Given the richness of its database, it would be tempting to invoke WordNet as the primary means of clarifying a keyword. This would be pointless, however, since the recipient agent has already gone to WordNet during the interpretation process.

Instead, the message source draws on the following methods of clarification:

- Locating synonyms in the source agent's ontology
- Providing a complete set of specializations (keyword as the union of its subclasses) from the source's ontology
- Providing a weak generalization from the source's ontology
- Providing a definition in formal logic—in particular, defining the keyword as the conjunction of other keywords

The interfaces to these clarification methods are formulated abstractly in the ontology API. It is up to the ontology to decide how to implement them.

4.3 Relevance Analysis

Relevance analysis is the process of evaluating the results of a query against the query itself. In our approach the query is specified via keywords, and the results are documents or URLs that are also described by keywords. So the problem reduces to evaluating how well these sets of keywords match. The evaluation is performed for each URL that is returned.

The current implementation computes a relevance measure for each result by accumulating *evidence of relevance* from the result document's keywords. Each keyword of the URL is examined in turn; if evidence of the keyword's relevance to the query can be found, the relevance measure is incremented.

Relevance analysis therefore reduces to the following question: given the set of *query* keywords, and a particular *result* keyword (of one of the returned URLs), what would constitute evidence that the result keyword is relevant to the query? If the result keyword *is* a query keyword, then clearly it is relevant. However, it is desirable to have other criteria since we are assuming an environment where there may not be a single controlling ontology. In the ontology API, the following tests are provided for determining whether keyword A and keyword B are relevant to each other:

- A is a specialization of B
- A and B have a common close generalization ("close" means not too far up the generalization tree)
- A and B have similar meanings (as decided by the ontology)
- A implies B
- The ranges of A and B intersect (i.e., A and B are compatible properties)
- A and B are connected by a series of facts that pair-wise have at least one predicate in common

The precise statement of the last criterion is rather involved; there are several variations, all of which contribute different degrees of relevance.

The API allows an ontology to support only a subset of the relevance criteria. In our current testbed we have implemented only the specialization and connectedness tests.

4.4 Ontology Evolution

The negotiation process culminates in one or both agents modifying their ontology to introduce a new

concept, a new distinction, or simply a new term for an existing concept. As in the case of explaining relevance, the algorithms and/or rules for modifying the ontology will depend on the ontology's representation and the tools used to maintain it. The best we can hope for in a general capability is an API that specifies (logically) the kinds of updates that may be performed. As shown in Figure 7, which is the Java definition of the API, the ontology evolution methods are inverses of the clarification and relevance evaluation (or explanation) methods.

Significant consistency issues arise in implementation of such methods. For example, a definition offered by agent A to agent B as clarification of a concept may be inconsistent with a definition already established in B's ontology. Depending on the nature of the concepts, the presence or absence of such inconsistencies may not be decidable or tractable.

```
public interface Ontology {

        // isWordNetOntology prevents duplicate calls to WordNet by
        // 1) interpret -> ontology and 2) interpret directly to WordNet
        public boolean isWordNetOntology();

        public String getName();
        public Vector getRelated(String term);
        public boolean isKnown(String term);

        // Clarification methods
        public TermTree getSynonyms(String term);
        public TermTree getConjunctiveDefinition(String term);
        public TermTree getDisjunctiveDefinition(String term);
        public TermTree getDefinition(String term);
        public TermTree getSlightGeneralization(String term);
        public TermTree getImportantCase(String term);

        // Relevance evaluation methods
        public boolean checkSpecializes (String term1, String term2);
        public boolean checkImplies (String term1, String term2);
        public boolean checkGeneralizesALittle (String term1, String term2);
        public boolean checkSimilarMeaning (String term1, String term2);
        public boolean checkRangesIntersect (String term1, String term2);
        public boolean checkConnected (String term1, String term2);

        public TermTree explainSpecializes (String term1, String term2);
        public TermTree explainImplies (String term1, String term2);
        public TermTree explainGeneralizesALittle (String term1, String term2);
        public TermTree explainSimilarMeaning (String term1, String term2);
        public TermTree explainRangesIntersect (String term1, String term2);
        public TermTree explainConnected (String term1, String term2);

        // Evolution methods (inverses of the clarification methods)
        public void putSynonyms(String term, TermTree t);
        public void putConjunctiveDefinition(String term, TermTree t);
        public void putDisjunctiveDefinition(String term, TermTree t);
        public void putDefinition(String term, TermTree t);
        public void putSlightGeneralization(String term, TermTree t);
        public void putImportantCase(String term, TermTree t);

        // Evolution methods (inverses of the relevance evaluation methods)
        public void putSpecializes (String term1, String term2);
        public void putImplies (String term1, String term2);
        public void putGeneralizesALittle (String term1, String term2);
        public void putSimilarMeaning (String term1, String term2);
        public void putRangesIntersect (String term1, String term2);
        public void putConnectedByFacts (String term1, String term2);

}
```

Figure7.: The Ontology Application Program Interface

Currently we sidestep such issues by accommodating alternative interpretations of a term. But this raises a host of questions, not only of how to represent such alternative meanings in an ontology, but more fundamentally how to assess the significance of the difference. For example, does it simply represent a different use of the term, or does it indicate a different set of beliefs about the term's referent? We have conjectured some tentative answers to such questions but have not yet implemented them.

4.5 The Ontology API

The API requires that an ontology communicate hierarchical information (such as a complex logical statement) in the form of a TermTree, which is a utility class provided with the API. For example, relevance explanations are all binary atomic propositions such as

```
Specializes (A, B)
GeneralizesALittle(A,B)
ConnectedByFacts(A,B)
```

where `A` and `B` are terms in the ontology. The corresponding TermTree has the binary predicate as its root, and the two terms as leaves.

Supporting a relevance explanation of the form ConnectedByFacts is an ExplanationOfFact (see Table 1), which is a list of the connecting facts. Each connecting fact is itself a logical proposition—that is, a combination of atomic propositions using the logical connectives `and`, `or`, `not`, etc. For an ontology-backed agent to support the API, it must be able to convert its internal representations into these TermTrees.

One way that we might lighten this requirement is to allow encoding in the standard Knowledge Interchange Format (KIF, 1998). This could be accomplished easily by providing an encoder and decoder between KIF expressions and TermTrees. The latter are more general, and are used for a variety of purposes within the ONP implementation.

5 CONCLUSION

We have described a novel approach to attacking the proliferation of information agent ontologies. We have created a software implementation framework that facilitates continued experimentation with the ONP. We have layered the reasoning process so that an ontology need only support a well-defined API in order to operate within the framework. We have demonstrated the utility of the approach in a testbed that includes ontology proxies for NASA's Global Change Master Directory and NOAA's Wind and Sea Index.

6 REFERENCES

Clark, P. and Porter, B. Building Concept Representations from Reusable Components. *AAAI 97*, pages 369 - 376.

Cybenko, G. and Jiang, G. Matching Conflicts: Functional Validation of Agents. *Agent Conflicts: Papers from the 1999 AAAI Workshop*. Technical Report WS-99-08, AAAI Press, Menlo Park, CA, 1999. Pages 14 - 19.

Decker, S., Brickley, D., Saarela, J., and Angele, J. A Query and Inference Service for RDF. Available at: http://www.w3.org/TandS/QL/QL98/pp/queryservice.html .

Fellbaum, C. WordNet: An Electronic Lexical Database. MIT Press, 1998.

Finin, T., Labrou, Y., and Mayfield, J. KQML as an Agent Communication Language , in Jeff Bradshaw (ed.), *Software Agents*, MIT Press, Cambridge, Available at http://www.cs.umbc.edu/kqml/papers. 1997.

FIPA Agent Communications Language, FIPA 97 Specification, Version 2. Available at: http://fipa.org/repository/index.html. October, 1998.

Heflin, J., Hendler, J., and Luke, S. Coping with Changing Ontologies in a Distributed Environment. In *AAAI-99 Workshop on Ontology Management*, AAAI/MIT Press, 1999.

KIF. Knowledge Interchange Format: Draft Proposed American National Standard, NCITS.T2/90-004. Available at: http://logic.stanford.edu/kif/dpans.html . 1998.

McCallum, A., Nigam, K., Rennie, J., and Seymore, K. Building Domain-Specific Search Engines with Machine Learning Techniques. *Intelligent Agents in Cyberspace: Papers from the 1999 AAAI Symposium*, March 22 - 24, Stanford, CA. Technical Report SS-99-03, AAAI Press, Menlo Park, CA, 1999. Pages 28 - 39

Takeda, H., Iwata, M., Sawada, A., and Nishida, T. An Ontology-based Cooperative Environment for Real-world Agents. *Second International Conference on Multi-Agent Systems (ICMAS 96)*. AAAI Press, 1996. Pages 353 - 360.