An Agent-Oriented Programming Language for Computing in Context

Renata Vieira[1], Álvaro F. Moreira[2], Rafael H. Bordini[3], and Jomi Hübner[4]

[1] Universidade do Vale do Rio dos Sinos
renata@exatas.unisinos.br

[2] Universidade Federal do Rio Grande do Sul
afmoreira@inf.ufrgs.br

[3] University of Durham
R.Bordini@durham.ac.uk

[4] Universidade Regional de Blumenau
jomi@inf.furb.br

Abstract. Context aware intelligent agents are key components in the development of pervasive systems. In this paper, we present an extension of a BDI programming language to support ontological reasoning and ontology-based speech act communication. These extensions were guided by the new requirements brought about by such emerging computing styles. These new features are essential for the development multi-agent systems with context awareness, given that ontologies have been widely pointed out as an appropriate way to model contexts.

1 Introduction

Context aware intelligent agents will be requeried if the vision of pervasive computing is to become real. Ontologies and ontology languages are becoming very popular as a way to model contexts, since they allow for the necessary reasoning to deal with the dynamic nature of this new computing style that is emerging. In other words, ontologies allow for different systems to come to a common understanding of the semantics of domains and services, and thus for the reuse of concepts in different contexts. In this complex new computing scenario, agent technologies, such as languages used to specify and implement them, have been left somewhat behind. In this paper, we present how we are extending an agent-oriented programming language for implementing agents that can operate in such environments. The AgentSpeak(L) programming language was introduced by Rao in [17]. The language was quite influential in the definition of other agent-oriented programming languages. AgentSpeak(L) is particularly interesting, in comparison to other agent-oriented languages, in that it retains the most important aspects of the BDI-based reactive planning systems on which it was based. Its relation to BDI logics [18] has been thoroughly studied [3] and a working

Please use the following format when citing this chapter:

Vieira, R., Moreira, Á.F., Bordini, R.H., Hübner, J., 2006, in IFIP International Federation for Information Processing, Volume 218, Professional Practice in Artificial Intelligence, eds. J. Debenham, (Boston: Springer), pp. 61–70.

interpreter for the language has been developed based on its formal operational semantics [15]. For AgentSpeak(L) to be useful in practice, various extensions to it have been proposed; from now on we refer to AgentSpeak(L) or any of its extensions generally as AgentSpeak. In particular, the formal semantics of an extended version of the language that allows for speech act-based communication, given in [16] has been used for the implementation of the *open source* interpreter ***Jason*** [1] [2]. Through speech act-based communication, an agent can share its internal state (beliefs, desires, intentions) with other agents, as well as it can influence other agents' states.

Despite the considerable improvement that has been achieved since the paradigm was first thought out [20], agent-oriented programming languages are still in their early stages of development and have clear shortfalls as far as their use in the software industry is concerned. One such shortfall of AgentSpeak has to do with the unrealistic simplicity of the way in which belief bases are implemented. A belief base that is simply an unstructured collection of ground predicates is just not good enough if we consider that mobile services and the semantic web make use of ontologies for representing knowledge (through more elaborate languages such as OWL, the Ontology Web Language). Due in part to this lack of structure, reasoning in AgentSpeak is limited to the unification mechanism applied to explicit knowledge. Another shortfall of AgentSpeak, that reduces its applicability for the development of semantic web multi-agent systems, is the absence of mechanisms to indicate the ontologies that have to be considered by agents in their reasoning. This shortfall imposes the assumption (that we had to adopt in [16], for instance) that all communicating AgentSpeak agents in a multi-agent application have a common understanding about terms that are used in the content of exchanged messages. This assumption is clearly unrealistic for semantic web applications as they typically require the integration of multiple ontologies about different domains.

The main contribution of this paper is to present an agent-oriented programming language which overcomes these limitations by bringing together: speech act-based inter-agent communication, improved descriptive and reasoning capabilities, and support for multiple ontology selection. The literature is abundant in proposals for speech-act-based communication and ontologies for the semantic web; we refer to [9, 23], just to mention a couple of examples.Pervasive, context-rich systems are being designed on the basis of the same apparatus [5]. Speech act theory is deeply related to BDI notions used in agent architectures, and in turn intelligent agents are also one of the key components in the pervasing computing vision. However, the design of such agents for such scenarios will require appropriate languages. To the best of our knowledge this is the first work aiming at integrating these technologies into a BDI agent-oriented programming language.

The paper is organized as follows: Section 2 gives a brief overview of the AgentSpeak language. Section 3 brings together ontologies and speech-act based communication in the AgentSpeak programming language. In the final section we draw some conclusions and discuss future work.

[1] `http://jason.sourceforge.net`

2 An Overview of AgentSpeak

The AgentSpeak programming language is an extension of logic programming for the BDI agent architecture, and provides an elegant framework for programming BDI agents. The BDI architecture is the predominant approach to the implementation of "intelligent" or "rational" agents [24].

$$
\begin{array}{lll}
ag & ::= & bs \quad ps \\
bs & ::= & at_1 \ldots at_n \qquad (n \geq 0) \\
at & ::= & P(t_1, \ldots t_n) \qquad (n \geq 0) \\
ps & ::= & p_1 \ldots p_n \qquad (n \geq 1) \\
p & ::= & te : ct \leftarrow h \\
te & ::= & +at \mid -at \mid +g \mid -g \\
ct & ::= & at \mid \neg at \mid ct \wedge ct \mid \mathsf{T} \\
h & ::= & a \mid g \mid u \mid h; h \\
g & ::= & !at \mid ?at \\
u & ::= & +at \mid -at
\end{array}
$$

Fig. 1. Syntax of AgentSpeak.

Figure 1 has the abstract syntax of AgentSpeak. An AgentSpeak agent is created by the specification of a set *bs* of base beliefs and a set *ps* of plans. In the original definition of the language an *initial set of beliefs* is just a collection of ground first order predicates. A plan is formed by a *triggering event* (denoting the purpose for that plan), followed by a conjunction of belief literals representing a *context*. The context must be a logical consequence of that agent's current beliefs for the plan to be *applicable*. The remainder of the plan is a sequence of basic actions or (sub)goals that the agent has to achieve (or test) when the plan, if applicable, is chosen for execution.

AgentSpeak distinguishes two types of goals: *achievement goals* and *test goals*. Achievement and test goals are predicates (as for beliefs) prefixed with operators '!' and '?' respectively. Achievement goals state that the agent wants to achieve a state of the world where the associated predicate is true. (In practice, these initiate the execution of *subplans*.) A *test goal* returns a unification for the associated predicate with one of the agent's beliefs; they fail otherwise. A *triggering event* defines which events may initiate the execution of a plan. An *event* can be internal, when a subgoal needs to be achieved, or external, when generated from belief updates as a result of perceiving the environment. There are two types of triggering events: those related to the *addition* ('+') and *deletion* ('−') of mental attitudes (beliefs or goals).

Plans refer to the *basic actions* (represented by the metavariable a in the grammar above) that an agent is able to perform on its environment. Such actions are also defined as first-order predicates, but with special predicate symbols (called action symbols) used to distinguish them from other predicates.

Consider a scenario where a tourist is walking around London's West End, planning his evening. The tourist's personal assistant can check for locally available plays

```
+concert(A,V,T) : likes(A)
   ← !book_tickets(A,V,T)

+!book_tickets(A,V,T) : ¬busy(T)
   ← call(V);
      ...;
      !choose_seats(A,V)
```

Fig. 2. Examples of AgentSpeak plans.

and concerts according to the user's preferences. Figure 2 shows some examples of AgentSpeak plans for this scenario. They tell us that, when a concert or play `A` is announced at venue `V` and `T`(so that, from the perception of the context, a belief `concert(A,V,T)` is *added*), then if this agent in fact likes artist `A`, then it will have the new goal of booking tickets for that concert. The second plan tells us that whenever this agent adopts the goal of booking tickets for `A`'s performance at `V`, if it is the case that the agent is not busy at T, according to his agenda, then it can execute a plan consisting of performing the basic action `call(V)` (assuming that it is an atomic action that the agent can perform) followed by a certain protocol for booking tickets (indicated by '...'), which in this case ends with the execution of a plan for choosing the seats for such performance at that particular venue.

3 AgentSpeak with Speech-Act Based Communication and Ontological Reasoning

3.1 Speech Act-Based Communication

As BDI theory is based on the philosophical literature on practical reasoning [4], agent communication in multi-agent systems is inspired by philosophical studies on the speech act theory, in particular the work by Austin [1] and Searle [19]. Speech act theory is based on the conception of language as action [1, 19]. In natural language, one has an illocutionary force associated to a utterance (or locutionary act) such as "the door is open" and another to a utterance "open the door". The former intends belief revision, whereas the latter intends a change in the plans of the hearer. When the theory is adapted to agent communication the illocutionary force is made explicit, to facilitate the computational processing of the communication act.

The Knowledge Query and Manipulation Language (KQML) [12] was the first practical communication language that included high level speech-act based communications. KQML "performatives" refer to illocutionary forces and they make explicit the agent intentions with a message being sent. The FIPA standard[2] for agent communication is conceptually similar to KQML, the differences being only in the sets of available performatives and a few other details.

[2] `http://www.fipa.org`

An operational semantics for AgentSpeak extended with speech-act based communication was given in [16]. That semantics tells exactly how the computational representation of Beliefs-Desires-Intentions of an agent are changed when it receives a message. It has also formed the basis for the implementation of AgentSpeak interpreters such the ***Jason*** [2].

Speech act based communication fits well with BDI agent oriented programming languages, such as AgentSpeak, since the semantics of both is given in terms of the agents mental states such as beliefs, desires and intentions. For this same reason, speech act based communication is an ideal approach for communication among agents in the web, in particular in applications where agents have to reasong about each other in order to negotiate and cooperate.

3.2 Ontologies and Ontological Reasoning

Developing applications that make full use of machine-readable knowledge sources as promised by the Semantic Web vision is attracting much of current research interest. More than that, the Semantic Web technology is also being used as the basis for other important trends in Computer Science such as Grid Computing [8] and Ubiquitous Computing [5]. Among the key components of the Semantic Web are *domain ontologies* [21]. They are responsible for the specification of the domain knowledge, and as they can be expressed logically, they can be the basis for sound reasoning in the specified domain. Several ontologies are being proposed for the development of specific applications [6, 14, 7, 22]. Description logics are at the core of widely known ontology languages, such as the Ontology Web Language (OWL) [13]. An extension of AgentSpeak with underlying automatic reasoning over ontologies expressed in such languages can have a major impact on the development of agents and multi-agent systems that can operate in a Semantic Web context. Although applications for the Semantic Web are already being developed, often based on the Agents paradigm, most such development is being done on a completely *ad hoc* fashion as far as agent-oriented programming is concerned.

In the Semantic Web framework, agents are responsible for making use of the available knowledge, autonomously interacting with other agents, so as to act on the user's best interest. Effective communication among these agents requires *(i)* a common understanding about the meaning of terms used in the content of messages, and *(ii)* communication abilities that can allow agents to know about each other in order to negotiate and cooperate. Agents achieve these requirements by sharing domain ontologies and by using speech-act based performatives in their communication. In the next two subsections we discuss the implications of extending the agent-oriented programming language AgentSpeak with these two features.

3.3 Ontologies and AgentSpeak

We start by presenting in Figure 3 the syntax of AgentSpeak with support for ontological reasoning. An agent consists of the specification *Ont* of the ontologies used by the agent, a set *bs* of belies, and a set *ps* of plans.

$$
\begin{array}{lll}
ag & ::= Ont \quad bs \quad ps & \\
 & & \\
Ont & ::= context\_ontology(url_1, \ldots url_n) & \\
 & & \\
bs & ::= at_1 \ldots at_n & (n \geq 0) \\
at & ::= C(t) \mid R(t_1, t_2) & \\
 & \quad \mid C(t)[s_1, \ldots, s_n; url] & (n \geq 0) \\
 & \quad \mid R(t_1, t_2)[s_1, \ldots, s_n; url] & (n \geq 0) \\
s & ::= \texttt{percept} \mid \texttt{self} \mid id & \\
 & & \\
ps & ::= p_1 \ldots p_n & (n \geq 1) \\
p & ::= te : ct \leftarrow h & \\
te & ::= +at \mid -at \mid +g \mid -g & \\
ct & ::= at \mid \neg at \mid ct \wedge ct \mid \mathsf{T} & \\
h & ::= a \mid g \mid u \mid h; h & \\
g & ::= !at \mid ?at & \\
u & ::= +at \mid -at &
\end{array}
$$

Fig. 3. Syntax of AgentSpeak with ontologies.

The specification of the ontologies is given by $context\_ontology([url_1, \ldots url_n])$, a special purpose predicate where each url_i is the URL of an ontology, usually described in the ontology language OWL. Each ontology consists of a set of class and property descriptions, and axioms establishing equivalence and subsumption relationships between classes (unary predicates) and properties (binary predicates). The belief base *bs* describes the state of an application domain by asserting that certain individuals are instances of certain classes and that certain individuals are related by a property. Each element of the belief base can be annotated with its source, which can be either a term identifying which was the agent in the society (id) that previously sent the information in a message, `self` to denote internal beliefs, or `percept` to indicate that the belief was acquired through perception of the environment. Beliefs can also be annotated with the *url* of the ontology where their associated classes and properties are defined. AgentSpeak plans are essentially the same as presented in Section 2

We now proceed to discuss the impact that ontological reasoning has on plan selection and querying in AgentSpeak programs. In $context\_ontology(url_1, \ldots url_n)$, the ontologies are written in some ontology language such as OWL, for simplicity though, the small extracts of ontologies we provide in this paper are expressed in a fragment of description logic instead of OWL.

Plan Selection A plan is considered relevant in relation to a triggering event if it has been written specifically to deal with that event or if it is a plan with a more general relevance that can be suitable for the triggering event. As an example let us consider the case of checking for plans that are relevant for a particular event and consider that the agent has at its disposal the following ontology:

$$atendee \equiv person \sqcap resgistered \sqcap \neg presenter$$
$$presenter \equiv speaker \sqcup paperAuthor \ldots$$

Suppose that a sensor in a smart meeting room somehow has detected in the environment the arrival of the speaker $john$. This causes the addition of the external event whose signaling term is $+speaker(john)$ to the set of events. Suppose also that $speaker \sqsubseteq presenter$ can be inferred from the ontologies specified in by the $context\_ontology(url_1, \ldots url_n)$. Under these circumstances, a plan with triggering event `+presenter(X)` is also considered relevant for dealing with the event. Observe that using subsumption instead of unification as the mechanism for searching for relevant plans potentially results in a larger set of plans. A plan is applicable if it is relevant and its context ct can be inferred from the ontologies and from the belief base. A plan's context is a conjunction of literals[3] $l_1, l_2, \ldots$. We can say that $Ont, bs \models l_1 \wedge \ldots \wedge l_n$ if, and only if, $Ont, bs \models l_i$ for $i = 1 \ldots n$.

Again, due to the fact that the ontologies and the belief base are structured and that reasoning is based on subsumption as well as instantiation, the resulting set of applicable plans might be larger. Suppose that plans with triggering events `+presenter(X)` and `+speaker(X)` were both considered relevant and applicable. The *least general* plan among those in the set of applicable plans should be selected plan. Here, the selected plan should be the one with triggering event `+speaker` as probably this plan has been written to deal more particularly with the case of invited speakers arriving in the room, rather then the more general plan which can be used for other types of presenters as well. On the other hand, if the particular plan for invited speaker is not applicable (e.g., because it involves alerting the session chair of the arrival of the celebrity speaker but the chair is not present), instead of the agent not acting at all for lack of applicable plans, the more general plan for speakers can then be tried, the relevance being determined by the underlying ontology instead.

Querying The evaluation of a test goal $?C(t)$ consists in testing if the formula $C(t)$ is a logical consequence of the agent's ontologies and the belief base. The crucial difference is that now the reasoning capabilities include subsumption instead of unification only which allows agents to infer knowledge that is implicit in the ontologies.

As an example suppose that the agent belief base does not refer to instances of $atendee$, but instead it has the facts $speaker(john)$ and $paperAuthor(mary)$. A test goal like $?atendee(X)$ succeeds in this case producing substitutions that map X to $john$ and $mary$.

Consistency of the Belief Base The belief base of an agent contains class assertions $C(t)$ and property assertions $R(t_1, \ldots t_n)$. The representation of such information should, of course, be consistent with the ontologies in $context\_ontology(url_1 \ldots url_k)$. Suppose for instance that by the ontologies it can be inferred that the concepts $chair$ and $bestPaperWinner$ are disjoint. Clearly if the belief base has that $chair(mary)$, the assertion $bestPaperWinner(mary)$ is not to be added to it, otherwise the belief base will become inconsistent.

[3] Note that in the context of the Semantic Web, open world is often assumed, so negation here is "strong negation", in the usual sense in logic programming.

Observe that the belief base of the agent can be updated by the addition of $chair(mary)[id; url]$, without the need for checking consistency as far as url is not a url in the set of url's specified in $context\_ontology(url_1 \ldots url_k)$. A belief like this expresses that $chair(mary)$ has been communicated by a certain agent id and that, when reasoning about it, the ontology in url should be used.

3.4 Ontologies and Communication

As usual in practice, we assume that the implementation of the AgentSpeak interpreter provides, as part of the overall agent architecture, a mechanism for receiving and sending messages asynchronously; messages are stored in a mail box and one of them is processed by the agent at the beginning of a reasoning cycle. Messages are sent by the execution of the action `.send` in the body of plans. The format of messages is $\langle mid, id, Ilf, at, url \rangle$, where mid is a unique message identifier; id is the identity of the agent to which the message is addressed to; Ilf is the illocutionary force associated with the message, at, the message content, is an atomic belief, and url is the ontology that must be used when reasoning about the message content at. For the purpose of this paper the interesting performatives are those whose semantics can lead to belief base modification or that involve ontological reasoning. Four of these performative are briefly described below.

`tell`: s informs r that the sentence in the message (i.e., the message content) is true of s — that is, the sentence is in the knowledge base of s (i.e., s believes that the content of the message is true);
`ask-if`: s wants to know if the content of the message is true for r;
`ask-all`: s wants all of r's answers to a question;
`ask-how`: s wants all of r's plans for a triggering event;

A *Tell* message might be sent to an agent either as a reply or as an inform action. Either way, before being added to the recipient belief base, the message content is annotated with the sender id and with the ontologies specified in the message. The receiver of an ask_if message will respond to the request for information. The answer should be given in relation to the ontology specified in the ask message. Note that ask_if and ask_all differ basically in the kind of request made to the receiver. With the former, the receiver should just confirm whether the received predicate (in the message content) follows from its belief base and the specified ontology; with the latter, the agent replies with all the predicates in the knowledge base (specified ontology plus belief base) that match the formula in the content of the message. The answer for an ask_how message is a set of all plans that are relevant for a triggering event which constitutes the message content. Note that these relevant plans are collected based on the subsumption relation defined in the ontologies specified in the message.

Ontologies have been considered a fundamental element since the first proposal of communication frameworks such as KQML. The reference to an ontology is a way to make sure that messages are interpreted in a previously agreed context between sender and speaker. From the performatives discussed above we can note that another benefit of combining speech-act based communication with ontologies is that the results

of message processing are more expressive, due mainly to the ontological reasoning based on the subsumption relations defined in ontologies.

4 Conclusions and Future Work

We have proposed an extension of the BDI agent-oriented programming language AgentSpeak that facilitates the development of agents that are able to communicate and reason about ontologies. These are important features for context aware agents. The AgentSpeak interpreter ***Jason*** [2] is currently being modified so that it can support the features discussed in this paper. ***Jason*** supports both closed and open-world assumption, and it is possible to run and debug the system in a distributed way over a network. Straightforward extensibility by user-defined internal actions, which are programmed in Java, is also available. To implement the AgentSpeak extension proposed here, ***Jason***'s inference engine needs to be extended to incorporate ontological reasoning, which of course can be done by existing software such as those presented in [10, 11]. We are currently considering the use of RACER [10] for extending ***Jason*** so that belief bases can also be written in OWL. The performatives tell, untell, achieve, and unachieve (amongst various others) have already been implemented. The practical usefulness of combining ontological reasoning and speech act based communication within an agent-oriented programming language for context computing seems quite clear, if we consider the increasing role of ontologies in the semantic web and pervasive computing. Although the semantic web and pervasive computing are still mostly visions for the future, with so much effort begin placed on this by the computer science community, these seem inescapable trends. As intelligent agents are central elements of both visions, languages with underlying ontological reasoning, as we proposed here, will be an important stepping stone towards consolidating those trends.

The language is suitable for the new scenario that pervasive computing applications are bringing about, agents are designed to act based on perceptions provided by the environments, such as the location of services and its availability. The services available may also be matched with specifications of users preferences and intentions.

Traditionally ontology-based systems are not multi-agent systems. The development of languages with support for ontological reasoning and multi-agent ontology based applications will require the use of techniques of ontology merging and ontology matching in order to allow heterogeneous agents to interoperate.

References

1. J. L. Austin. *How to Do Things with Words.* Oxford University Press, London, 1962.
2. R. H. Bordini, J. F. Hübner, and R. Vieira. ***Jason*** and the Golden Fleece of agent-oriented programming. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming*, chapter 1. Springer, 2005.
3. R. H. Bordini and Á. F. Moreira. Proving BDI properties of agent-oriented programming languages: The asymmetry thesis principles in AgentSpeak(L). *Annals of Mathematics and*

Artificial Intelligence, 42(1–3):197–226, Sept. 2004. Special Issue on Computational Logic in Multi-Agent Systems.
4. M. E. Bratman. *Intentions, Plans and Practical Reason*. Harvard University Press, Cambridge, MA, 1987.
5. H. Chen, T. Finin, A. Joshi, F. Perich, D. Chakraborty, , and L. Kagal. Intelligent agents meet the semantic web in smart spaces. *IEEE Internet Computing*, 19(5):69–79, 2004.
6. H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. In *International Conference on Mobile and Ubiquitous Systems: Networking and Services*, Boston, MA, August 2004.
7. Y. Ding, D. Fensel, M. C. A. Klein, B. Omelayenko, and E. Schulten. The role of ontologies in ecommerce. In Staab and Studer [21], pages 593–616.
8. I. Foster and C. Kesselman, editors. *The Grid 2: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, second edition, 2003.
9. N. Gibbins, S. Harris, and N. Shadbolt. Agent-based semantic web services. *J. Web Sem.*, 1(2):141–154, 2004.
10. V. Haarslev and R. Moller. Description of the RACER system and its applications. In *Proceedings of the International Workshop in Description Logics 2001 (DL'01)*, 2001.
11. I. Horrocks. FaCT and iFaCT. In *Proceedings of the International Workshop on Description Logics (DL'99)*, pages 133–135, 1999.
12. Y. Labrou and T. Finin. A semantics approach for KQML—a general purpose communication language for software agents. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*. ACM Press, Nov. 1994.
13. D. L. McGuinness and F. van Harmelen, editors. *OWL Web Ontology Language overview. W3C Recommendation*. Avalilable at http://www.w3.org/TR/owl-features/, February 2004.
14. S. E. Middleton, D. D. Roure, and N. R. Shadbolt. Ontology-based recommender systems. In Staab and Studer [21], pages 577–498.
15. Á. F. Moreira and R. H. Bordini. An operational semantics for a BDI agent-oriented programming language. *Proceedings of the Workshop on Logics for Agent-Based Systems (LABS-02)*, pages 45–59, 2002.
16. Á. F. Moreira, R. Vieira, and R. H. Bordini. Extending the operational semantics of a BDI agent-oriented programming language for introducing speech-act based communication. In *Declarative Agent Languages and Technologies, Proceedings of the First International Workshop (DALT-03)*, LNAI, pages 135–154, Berlin, 2004. Springer-Verlag.
17. A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computable language. In *Proceedings of the Seventh Workshop on Modelling Autonomous Agents in a Multi-Agent World (MAAMAW'96), 22–25 January, Eindhoven, The Netherlands*, number 1038 in LNAI, pages 42–55, London, 1996. Springer-Verlag.
18. A. S. Rao and M. P. Georgeff. Decision procedures for BDI logics. *Journal of Logic and Computation*, 8(3):293–343, 1998.
19. J. R. Searle. *Speech Acts: An Essay in the Philosophy of Language*. Cambridge University Press, Cambridge, 1969.
20. Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60:51–92, 1993.
21. S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.
22. R. Stevens, C. Wroe, P. W. Lord, and C. A. Goble. Ontologies in bioinformatics. In Staab and Studer [21], pages 635–658.
23. M. Uschold. Where are the semantics in the semantic web? *AI Magazine*, 24(3):25–36, 2003.
24. M. Wooldridge. *Reasoning about Rational Agents*. The MIT Press, Cambridge, MA, 2000.