Preface

Here come the golden words.

Place, Month Year *First Name, Surname*

Table of Contents

1. Ontologies in Agent Architectures

Katia Sycara and Massimo Paolucci

Carnegie Mellon University, Pittsburgh PA 15213, USA
email: katia,paolucci@cs.cmu.edu

1.1 Introduction

Agents display a dual behavior: on the one hand they are goal directed programs that autonomously and proactively solve problems for their users; on the other hand agents have a social dimension when they interoperate as part of Multi-Agent Systems (MAS). As autonomous problem solvers, agents need to develop a model of their environment that allows them to reason on how the actions that they perform affect their environment, and how those changes lead them to achieve their goals. Ontologies provide the conceptual framework that allows agents to construct such models: ontologies describe the type of entities that agents encounter, the properties of those entities, and the relations between them. For example, a stock reporting agent may require ontologies describing not only concepts like ticker symbol, but also the relation between stock and ticker symbol and properties of stocks such as its value expressed in some currency.

In their social dimension, agents in a MAS necessarily interact with other agents. They may compete for resources, or collaborate toward the solution of a problem or toward the achievement of a common goal. While some of these interactions are quite accidental, for example when an agent has to wait to access a resource because other agents use it, the agents' main tool for interaction and interoperations is communication. Communication allows agents to exchange information as well as requests for services. For example an agent that needs to make financial decisions may want to ask other agents for stock quotes, or it may want to subcontract parts of its financial analysis to other agents. When combining communication and problem solving, the solution of a problem is rarely restricted to the agent itself, rather it often involves other agents. Ontologies provide agents the basic representation that allows them to reason about those interactions, but also, and most importantly, ontologies provide agents with shared knowledge that they can use to communicate and work together.

In order to interact, agents must first know of each others' presence and location in the MAS. Since MAS are open systems, i.e. they are societies where agents enter or leave at unpredictable times, any attempt at programming the agents under the assumption that they know of all their peers would fail, or reduce the MAS to a closed system whose capabilities cannot be modified without re-programming all the agents and their interactions. The alternative is to introduce a *discovery* mechanism so that agents can find each other

dynamically. In such a discovery scheme, agents entering the system advertise their presence, while agents in need of services, issue service requests. The objective of the discovery process is to locate those agents whose advertisement matches an issued request. Dynamic discovery removes the need for hardcoded references, while allowing agents to enter and exit the MAS and engage in flexible global interaction patterns.

The dual behavior, individual problem solving and social interactions, displayed by agents is reflected in different approaches toward ontologies. In general, we can distinguish between *private* ontologies that allow the agent to organize its own problem solving and reasoning, and *public* ontologies that the agent shares with the rest of the agents in the MAS.

Private ontologies serve the problem solving purposes of the agent. Because of this function it is often difficult to distinguish between ontologies as conceptualization of the domain of the agent, and the knowledge representation needed by the problem solving process that the agent employs. For example, an agent that uses planning technology to solve its problems may base its private ontologies on a STRIP [1.13, 1.17] like representations, with the result of representing its domain in terms of actions that the agent can perform, and plans that the agent constructs. On the other hand, an agent based on a DATALOG Data Base would instead represent its domain in terms of static facts and inference rules that allow the agent to extract the knowledge that is implicitly stored in the data base

Public ontologies are shared among multiple agents in the MAS. They are independent of the specific problem solving mechanism adopted by the agents; furthermore, ontologies need to respect the heterogeneity of the agents in the MAS. For example, while each agent in the MAS may entertain its own private knowledge representation, agents may also share knowledge using ontology languages, such as DAML+OIL [1.8], which provide their own formalism, and their own independent proof theory. The main role of public ontologies is to support agents in their interoperation; particularly in communication and information exchanges. Specifically, public ontologies need to provide a description of the domain of the whole MAS that is shared across all the agents, and a shared vocabulary so that agents understand the content of the messages that they exchange.

A crucial subset of the domain of an agent is the MAS the agent is part of. Other agents in the MAS affect what the agent does and how it does it. To this extent, ontologies should support the description of agents in terms of their capabilities, basic information on how to contact them, interaction protocol, reliability, reputation, security and so on. Furthermore, the social dimension of a MAS does not emerge just by developing agents and hoping that somehow they interact; rather agents need an infrastructure that provides the services such as location registries, and conventions such as standard protocols that allow agents to find each other and interact [1.49]. Therefore,

ontologies should also support the description of the MAS infrastructure, what kind of registries it employs [1.54], what kind of protocols, and so on.

It is virtually impossible to find a common denominator across all possible private agent ontologies. This is because intelligent agents have been based on virtually every problem solving mechanism that has been invented in AI. Examples range from deductive agents based on ConGolog [1.18] that use ontologies expressed in Situation calculus [1.28], to agents exploiting various types of synthetic planning, as for example the RETSINA agents [1.46, 1.34] based on HTN planning [1.12], or BDI [1.36] inspired agents based on reactive planning [1.16, 1.24, 1.32], to agents that use decision theoretic and game theoretic approaches [1.26, 1.39].

In this paper we concentrate on the use of public ontologies in agent discovery, interoperation and communication. The rest of the chapter is organized as follows: section 1.2 concentrates on the contribution of ontologies to agents' social activities. We will show that ontologies are essential to the description of agents' "social interfaces", and in particular the description of agents' capabilities which are needed for agent discovery in a MAS. In section 1.3 we describe ontologies for agent communication and show how ontologies contribute to mutual understanding of agent messages and interactions. Section 1.4 We will then conclude highlighting some of the open challenges.

1.2 Agent Discovery

Dynamic discovery mechanisms have four distinctive characteristics: the first one is a representation of the agents in the system; the second one a matching process that identifies the similarities between requests for agents, and advertisements of agents in the MAS; the third characteristic is a set of infrastructure components such as registries and protocols that support discovery; and finally, the last characteristics of discovery processes, relates to the problem solving of the agents. The different discovery schemes proposed in the literature are distinguished by the way they address the four different aspects of discovery, leading to a range of various levels of flexibility in the dynamic reconfiguration and coordination regimes of the agents in the MAS. For example, OAA [1.31] assumes a centralized broker which mediates all the interactions between agents; Contract Net [1.43] does not assume any global registry, rather agents use a bidding protocol to reply to task announcements and contract tasks.

Ontologies are an essential ingredient of discovery. They provide the means to represent the different aspects of agents and the basic mechanisms for the match between agents requests and agents advertisements. First of all, since agents are objects in the domain, any ontological representation of the domain should necessarily also represent the agents themselves. But more importantly, agents modify their environment, so any descriptions of an agent necessarily refers to the ontological description of the environment the agent

lives in. In this section, we analyze the different aspects of discovery, and its relations with ontologies. Specifically we will analyze the representation of agents in advertisements and requests, and how this representation affects the matching process and the agent's problem solving.

1.2.1 Capability Representation

Agents can be represented at many levels of abstraction. At the physical level they are characterized by their ports and network protocols. This is essentially the representation provided by the Web services infrastructure specifications such as WSDL [1.6] and UDDI [1.51]. At a more abstract level, agents interact with other agents through a communication language, and use a set of ontologies to encode their messages and interpret the messages that they receive. From another point of view, agents are characterized by their capabilities, their interaction protocol, the problem solving procedures that they employ, the legal entity that is responsible for their correct functioning, and so on.

In this section we concentrate on the representation of capabilities which we believe is crucial for the discovery of autonomous intelligent agents in an open MAS. In such systems, agents are aware of the problem solving needs that emerges during their reasoning, but they do not know which agent in the MAS can satisfy those needs. The task of the agent is then to abstract from the specific problem to the capabilities that it expects from a provider. For example, an agent that provides financial advice may need the latest quote of the IBM stock. To this extent, the agent should transform the particular problem, i.e. get the quotes of the IBM stock, to a description of the capabilities it expects from the stock quotes provider, i.e. stock market reporting. Finally, the financial planning agent should use that capability description to locate the stock reporting agents.

A number of capability representation schemes have been proposed by the agent community and more recently by the Web Services and Semantic Web community. Each of these representation schemes makes different assumptions on the representation of the agents, and most importantly on what kind of ontologies they use in their representation. Specifically, we distinguish between two types of representation schemes: the first one assumes ontologies that provide an *explicit* representation of the tasks performed by agents. In those ontologies, each task is described by a different concept, while agents are described by enumerating the tasks that they perform. The second representation scheme describes agents by the state transformation that they produce, therefore there is no mention of the task performed by the agent; the task is *implicitly* represented by the state transformation of the agent's inputs to the outputs it produces.

The two approaches to agent representation provide two ways to use ontologies. The schemes that make an explicit use of tasks ontologies provide a

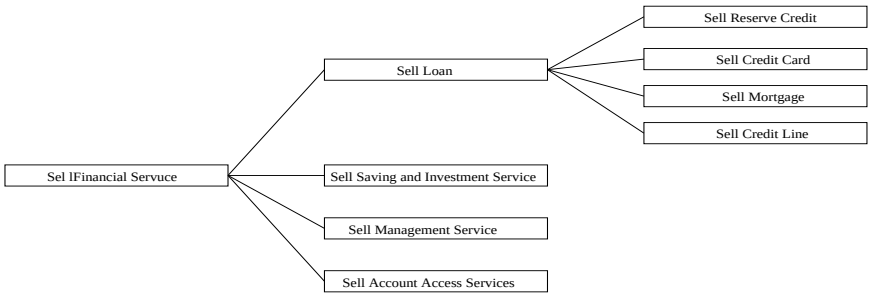

Fig. 1.1. Fragment of ontology of loan selling tasks

straightforward way to locate agents with give capabilities. But they also require ontologies that assign a concept for each task performed by each agent in the MAS. Since agents can perform many different tasks, these ontologies can grow very large thus becoming unmanageable and may not scale up when agents with new capabilities enter the system. Another drawback of the explicit representation schemes is that they do not represent what information the provider agent needs in order to interact with a requesting agent. The implicit representation schemes require only concepts that describe the domain of the agent, and then use those concepts to describe the task computed by the agent. In addition, some implicit representation schemes provide information (e.g. in terms of needed inputs) that a requester must provide to interact with a service provider agent. On the other hand, explicit representations facilitate the matching process since there is no need to infer the task from its implicit representation. Each capability representation scheme strikes a different balance between the two extremes depending on the ontologies that it has available, and how closely they describe the capabilities of the agents in the MAS.

1.2.2 Explicit Agent Capability Representations

An example of ontology which provides an explicit description of tasks and processes is the MIT Process Handbook [1.30]. Figure 1.1 shows a fragment of the specialization hierarchy of the ontology of tasks with the root "sell financial services" [1.30]. Furthermore, it shows that the concept `Sell Loan` is a specialization of `Sell Financial Service` which in turn is specialized by `Sell Credit Card`, `Sell Mortgage` and other concepts. In turn, the ontology associates to each process properties such as `port` that describes the I/O behavior of the process, and `decomposition` that describes how the process is realized by the composition of other processes described in the ontology. The MIT Process Handbook can be used to index Web services and agents for later retrieval [1.1]. For example, an agent that sells loans would be associated with the concept `Sell Loan` in the taxonomy in figure1.1. The advantage of this approach is that it reduces the burden of modeling the agent, since it

is represented by the task it performs. The disadvantage of this representation, at least in principle, is that it is impossible to distinguish between Web services or agents that sell loans whose amount is greater that $50,000 from agents that sell loans whose amount is smaller than $10,000[1]. To represent these constraints on the loan amount that the two agents offer would require, at least in principle, the definition of two subclasses of `Sell Loan` to describe the two different cases.

The use of explicit task ontologies for agent representation has been proposed for a number of agent representations and more recently for representation of semantic web services. InfoSleuth [1.33] describes different aspects of agents using concepts drawn from different ontologies. An agent representation based on InfoSleuth is shown in figure 1.2. The representation inherits from the `finance` ontology the definition of stock quotes, and from the `conversation` ontology the specifications of the communication language that the agent uses, namely KQML. Requests for capabilities and advertisements of capabilities have the same representation; a match between an advertisement and a request is recognized when the advertised agent has all the features that are specified in the request.

```
capability example-capability
   ontology finance
     class stockQuote
       slot ticker
       slot delay in immediate
   ontology conversation
     class conversation
       slot language in kqml
```

Fig. 1.2. example of stock quote agent in InfoSleuth

The representation used by InfoSleuth takes advantage of the types of agents that were used in InfoSleuth applications, namely information agents. For example, from the description of the agent above there is no way of telling whether the agent provides information about conversation languages, or whether it uses a specific conversation language. In both cases the description is exactly the same. The resolution of the ambiguity is possible only with the reference to the context since no agent in InfoSleuth provides information about communication languages.

Other capability representation languages include Phosphorous [1.19, 1.45] which represents capabilities as a predicate in which the functor is the name of the task represented and the arguments are the parameters of

[1] We selected "amount" because it does not seem to be represented in the ontology. In general any unmodeled feature can be used here instead.

the task. The matching process is reduced to a series of tests that analyze the subsumption relation (or the inverted subsumption) between the advertisement and the request. Another approach that makes use of the semantic web has been proposed in [1.50]; tasks of web services are represented by the instance of the task that they perform, and a set of properties that depend on the task. Matching of the request with the advertisement is reduced to a subsumption or inverted subsumption between the advertisements and the request.

1.2.3 Implicit Agent Capability Representation

LARKS [1.48, 1.47] provides an example of agent representation that uses an implicit task definition. LARKS represents the capabilities of agents as transformation from a set of inputs to a set of outputs. Consistent with this view, each LARKS advertisement and request requires the specification of seven features of the agent.

Context The context, or the domain, of the agent.
InTypes and ***OutTypes*** An optional set of input and output data types.
Inputs and ***Outputs*** The inputs that the agent expects, and the outputs that it generates.
InConstraints and ***OutConstraints*** Sets of constraints on the inputs and outputs if the agent.
ConcDescription a list of concepts used in the agent description, that do not appear in the ontologies.

Different agents are described by different settings of the parameters. For example, a stock quote reporting agent in LARKS is described in figure 1.3; the context specifies that the agent operates in the domain of Finance and that given a ticker as input, it generates a quote as output. This definition of course assumes that there exist ontologies that define the concepts of Finance, ticker, and quote and that those ontologies are shared by the agent that provides the service, all possible requesters and the infrastructure components that are responsible for the matching between the two.

```
        Context =  Finance
         InTypes =  String
        OutTypes =  Real
          Inputs =  ticker
         Outputs =  quote
    InConstraints =  retail ∧ banking
   OutConstraints =  retail ∧ banking
  ConcDescription =
```

Fig. 1.3. LARKS advertisement of stock reporting agent

The matching process adopted by LARKS uses a combination of information retrieval techniques such as TF/IDF [1.38] and logic inference to identify the relation between the advertisement and the request. The information retrieval techniques are used as heuristics that remove all the advertisements that for sure have nothing to do with the request. The logic inference is based on subsumption and on weighted relations to compute the distance between the concepts used in the advertisement and the concepts used in the request.

The LARKS matching process allows for *partial matches* to find advertisements of providers whose capabilities are *similar enough* to the capabilities requested. This is because in practice, it is unrealistic to expect that the request will match exactly one or more advertisements of agents in the MAS since the requester and provider may use local ontologies that could be different. The goal of partial matching is to maintain validity by rejecting matches of advertisements and requests that are totally unrelated, and measuring the degree of match between two capability descriptions: the more similar the services described, the higher the degree of match. LARKS deals with partial matching by measuring the conceptual distance between the advertisement and the request, and it leaves to the requester to determine a cut-off value that limits the distance between the advertisement and the request to some value acceptable to the requester. A high cut-off value would eliminate most of the providers even if their capabilities are close to the capabilities requested, a low value would include many providers whose capabilities may differ substantially from the capabilities requested.

1.2.4 Combining Explicit and Implicit Representation

Other approaches to capability representation use a combination of the implicit and explicit representations. These approaches combine the ease of use of ontologies of tasks, when those ontologies are present, with the ability of describing the capability of any agent provided by the implicit approach.

The most notable example of this approach is the DAML-S Service Profile[2] [1.2]. It provides an implicit description of agents in terms of a host of features as LARKS does. In addition profiles are concepts in a DAML ontology and therefore it is possible to organize them in a taxonomy similar to the taxonomy shown in figure 1.1.

DAML-S Profiles provide three types of agent information: the first type is a specification of an *actor* who is responsible for the advertisement or the request. The second type of information consists of a host of *Functional Attributes* such as the category of service, quality of service and additional service parameters which can be used to specify details of the service. The

[2] DAML-S is composed of three modules: the Service Profile that describes the agent, the Process Model that describes the workflow of the agent and the Service Grounding that describes the communication details of how to interact with the agent.

last type of information specifies the *Functional Specification* which defines the transformation produced by the agent in terms of the inputs, outputs, preconditions and effects. Functional Attributes and the Functional Specification describe the capability of the agent.

Functional Attributes allow to specify features of the agent that are important to decide whether the agent adequately addresses a request. For example, one of the functional attributes of a stock reporting agent may be the delay in stock reporting with respect to the market. This could be described using DAML-S by specifying a service parameter as shown in figure 1.4. In the figure, the value of the stock quoting delay is 1 minute.

```
<profile:serviceParameter>
   <profile:QuoteDelay rdf:ID="agentDelay">
     <profile:serviceParameterName rdf':value=''one minute"/>
     <profile:sParameter rdf:resource="time:#1minute"/>
   </profile:GeographicRadius>
</profile:serviceParameter>
```

Fig. 1.4. example of DAML-S service parameter

The functional representation describes the transformation computed by the agent in a way that is similar to LARKS. An example of functional representation is shown in figure 1.5. The agent requires as input a ticker symbol and it returns the quote of the stock. To execute correctly the requester will have to prove that it is a valid subscriber to the service (we assume in this example a subscription-based payment model for the stock quoting service). As a result of invoking the service, the requester's account will be charged[3]

```
        Inputs =  ticker
       Outputs =  quote
 Preconditions =  valid(membership)
       Effects =  charged(account)
```

Fig. 1.5. DAML-S functional representation of stock agent

The matching process for DAML-S [1.35] recognizes a match between the advertisement and the request, in case when the advertised service could be used in place of the requested service. Specifically, the matching process uses the inputs and outputs, of the advertisement and the request to derive the different degrees of match: *exact* when the advertisement and the request are

[3] At the time of writing, DAML does not support a rule language, therefore, conditions like valid(membership) or charged(account) cannot be expressed.

synonymous, *plugIn* when the advertisement subsumes the request, *subsumed* when the request subsumes the advertisement, and *fail* when no relation can be found. When plugIn match holds, the provider claims it provides all the requester needs, but the advertisement may be so generic that it is useless for the requester. For example, a provider may advertise that it retails all kinds of products. This advertisement partially matches a request for a book seller. When subsumed match holds, the provider provides only part of the capabilities that the requester needs, and it is up to the requester to verify if those capabilities are enough. For example, a provider may advertise that it sells Ford and Chrysler cars. This advertisement matches in a subsumed match a request for a provider of sedan cars. Presumably the provider provides Ford and Chrysler sedans, and it is up to the requester to decide whether these brands of sedans meet its needs so it would interact further with the provider.

As pointed out above the DAML-S Profile specifies a class in a DAML ontology, therefore, at least in principle it is possible to follow a different matching process that makes use of DAML classifiers to match advertisements and requests. The challenge with matchmaking of DAML-S Profiles is to combine the explicit and implicit way of matching to improve the final result.

1.2.5 Trade-offs

We have described how the capabilities of agents play an essential role in discovery, and we analyzed in some detail different agent and web service representation formalisms. We classified them into two main types: explicit, when they take advantage of an extensive taxonomy of tasks, and implicit, when such an ontology is not used. There are clearly trade-offs between the two representations. Representations that use explicit representations of tasks require ontologies that are difficult to built. Furthermore, they suffer from the weakness that every time an agent with a new functionality enters the MAS, a new class must be defined in the ontology. Implicit representations do not need any explicit coding of tasks in the ontology providing a natural expression of tasks using domain ontologies already available. The tradeoff is that the use of explicit task representations makes matching of requests and advertisements much easier reducing it to test what kind of subsumption relation holds between advertisements and requests. Matching between capabilities expressed implicitly is more complex. Since there may not be an easy way to classify different tasks, matching requires a careful comparison between the input/output transformation described in the request with the transformations described in the advertisements.

1.3 Ontologies for Agent Communication

In a MAS, knowledge and capabilities are distributed across the agents in such a way that no single agent has a complete knowledge of the whole

multiagent system and no single agent can perform all the operations that can be performed by all the other agents. Despite their limited knowledge and capabilities, agents can ask other agents to perform some action or to provide information. Ultimately, communication allows any agent to extend the set of goals that it can satisfy by asking other agents with a different set of capabilities to contribute to its plans. For example an agent in the blocks world may be able to slide boxes, but not lift them; nevertheless, it may still construct plans for stacking blocks by pushing the blocks close to each other and then asking a lifting agent to put them one on top of the other.

The ability to communicate with other agents is one of the central skills of any agent in a Multi-agent system. Furthermore, since communication extends the knowledge and the capabilities of any agent, it becomes an integral part of the problem solving of any agent. Crucially, in the example of the blocks world agent described above, the resulting plan contains both physical actions: pushing blocks together, and communicative actions: asking the lifting agent to stack blocks.

The interleaving of communication and problem solving requires the agent to decide whether to achieve its goals via direct action, or by asking other agents to achieve some of those goals. In turn, this requires communication actions to be described as any other actions: they must have preconditions and they achieve some effect. This is the approach taken by the speech act theory [1.3, 1.41, 1.7], which, in a nutshell, adopts the view that by saying something the speaker achieves the effect of producing either a change in its world, or a change in the beliefs and intentions of the listener. For example, some utterances have very striking effects: when a public official declares two people married, the effect is a change in their marital status. Other utterances have more subtle effects, for instance by saying "its raining" the speaker attempts to make the listener belief that indeed it is raining; similarly, by asking the listener to do something, the speaker may induce in the listener the desire to do what was asked.

1.3.1 Representation of Speech Acts

Speech act theory, and its predication that communicative actions are essentially equivalent to other action in the world proved to be very appealing to the agent community. First, it allows a precise definition of communication actions and of the consequences of message exchanges; second, communicative actions can be defined in terms of planning operators and they may easily inserted in the plan that the agent is developing to achieve its goals[4].

[4] The linguistic basis of the speech act theory is more complicated than described here, and cannot be represented faithfully with planning operators. Nevertheless, our description is faithful to the use that agents make of it.

Speech act theory is the base of ontologies of communication actions such as KQML [1.14] and FIPA ACL[5] [1.15]. For example FIPA ACL specifies 22 speech acts, which are described by two properties: the first one specifies *feasibility preconditions* which are conditions that should be satisfied for the communicative action to be performed; and *rational effects* that represent the effects that the speaker hopes to achieve with the communicative action. As an example, consider the definition of the *inform* action in FIPA ACL as shown in figure 1.6. It states that it is appropriate for the speaker to inform the listener of ϕ when (feasibility condition: FP) the speaker believes the ϕ holds, and it does not believe that the listener believes that ϕ is true, nor the speaker is uncertain on whether the listener believes ϕ to be true. The result (rational effect) of the inform action is that the listener believes ϕ.

Inform
$< speaker, inform(listener, \phi) >$
FP: $B_{speaker}\phi \vee \neg B_{speaker}(B_{speaker} f_{listener}\phi \wedge U_{speaker} f_{listener}\phi)$
RE: $B_{listener}\phi$

Fig. 1.6. FIPA definition of Inform

The use of FIPA ACL allows the speaker agent to have expectations on the behavior of the listener, but there are no guarantees that the expectation will be met by reality. The first problem is that since the listener is an autonomous agent with its own knowledge, it may already know that ϕ is false or it may believe that the speaker is a liar and therefore it may reject the inform on those grounds. The second problem is that both FIPA ACL and the speech acts theory assume that the listener understands ϕ, but when this assumption does not hold, the listener will fail to hold the beliefs that the speaker expects it to hold. Suppose for example that the speaker informs the listener of the IBM quotes using the message in figure 1.7. Even if we assume that both parties agree on the FIPA ACL interpretation of inform speech act, still the listener has to understand the meaning of the terms "quote", "IBM," and "$87" and the way their meanings are combined to form the meaning of the statement. Furthermore, the speaker should agree with the listener's interpretation. The failure to reach an agreement on the meaning of those terms would lead to a failure to share the same interpretation of the message and it would prevent successful communication between the speaker and the

[5] KQML and FIPA ACL are usually described as agent communication languages, not as ontologies. In reality they are both, they provide a syntax that can be used by agents to compile messages, and a semantics that specifies concepts like message and speech act. To this extent FIPA ACL and KQML are effectively ontologies for agent communication.

listener. This very simple example shows that in the absence of any shared ontologies that define the meaning of the terms used in the message exchange, the two parties have no way to communicate.

```
(inform
   :sender speaker
   :receiver listener
   :content (quote IBM $87))
```

Fig. 1.7. example of inform message

Ontologies provide the tools to interpret the content of the message. For example, the speaker may encode its message using the DAML+OIL ontology shown in figure 1.8. The ontology provides a description of *quote*, specify that it relates to the concept *Ticker* through the ticker relation and to a *Value* through the *hasValue* relation. Furthermore, it specifies how these concepts are related to other concepts that may be defined elsewhere. For example, the ticker symbol is related to a company description that will contain other information about that company, while the value relates to a currency and so on. In this framework, the content of the message could be expressed as an instance built on the basis of the ontology. In this case it describes a quote, using an instance of the ticker that happens to point to IBM, and the value of the quote is 89 measured in US Dollars.

As the example shows, ontologies, by providing a shared conceptualization of the domain, effectively contribute to agent communication by providing a language and dictionary that can be used to express concepts and statements about the domain of the agents. Furthermore, those languages and dictionaries are standardized and shared by all the agents in the MAS. While each agent may have its own knowledge and its own beliefs on the value of the IBM stocks, they all share the same understanding of terms like IBM and quote so that all of them achieve the same interpretation of the statements that are communicated.

The second contribution of ontologies is in the use of the knowledge that the agent receives through its communication with other agents. The rational effect of the inform speech act, as we defined it in figure 1.6, is that the listener believes the content of the message. This means that the content of the message has to be added to the knowledge base of the agent and used to derive new inferences. Ontologies support this integration by providing the bases for the organization of knowledge in the agents knowledge base and of a proof theory so that the agent can derive inferences from the knowledge it gathers.

Message Content

```
<Quote>
  <hasTicker>
    <Ticker>
      <symbol
        rdf:value="IBM"/>
      <company
        rdf:resource="#IBM"/>
    </Ticker>
  </hasTicker>
  <hasQuote>
    <Value>
      <currency
        rdf:resurce="#US_Dollar"/>
      <hasValue rdf:value="89"/>
    </Value>
  </hasQuote>
</Quote>
```

Ontology

```
<class rdf:ID="Ticker">
  <property rdf:ID="symbol"/>
     <domain rdf:resurce="#Ticker"/>
     <range rdf:resurce="xsd#string"/>
  </property>
  <property rdf:ID="company"/>
    <domain rdf:resurce="#Ticker"/>
    <range rdf:resurce="#Company"/>
  </property>
< /class>

<class rdf:ID="Value">
  <property rdf:ID="currency"/>
    <domain rdf:resurce="#Value"/>
    <range rdf:resurce="#Currency"/>
  </property>
  <property rdf:ID="hasValue"/>
     <domain rdf:resurce="#Value"/>
     <range rdf:resurce="xsd#float"/>
  </property>
< /class>

<class rdf:ID="Quote">
  <property rdf:ID="hasTicker"/>
    <domain rdf:resurce="#Quote"/>
    <range rdf:resurce="#Ticker"/>
  </property>
  <property rdf:ID="quote"/>
    <domain rdf:resurce="#Quote"/>
    <range rdf:resurce="#Value"/>
  </property>
< /class>
```

Fig. 1.8. Description of a Quote using a DAML+OIL ontology.

1.3.2 Extending communication to conversations

The use of speech acts in communication is quite limited. It provides a framework for interpreting each message that agents exchange, but it does not provide a framework that extends to longer conversations. FIPA ACL, for example, mandates that every `query` action should be responded to with an `inform` action. But the use of FIPA ACL speech acts does not extend beyond these two message exchanges. Ideally, long conversations should be built as composition of simple atomic speech acts as human conversations are. From this point of view, agent A may ask B do X, B may query A asking for clarifications, A responds to B, and B satisfied commits to X. While humans find

very easy to be part of this kind of conversations, software agents are not yet sophisticated enough to be part of the "game of language" [1.29].

The management of long conversations suffers from a number of serious problems. The first problem is that the compositional semantics of speech acts may result in multiple interpretations for the same sequence of speech acts. This is the so called *basic problem* [1.20] which may lead to ambiguity and misunderstanding between the different parties. The second problem is in the definition of speech acts in FIPA ACL: speech acts require the speaker to have a model of the private beliefs of the listener which are never available, so the very preconditions of FIPA ACL speech acts, if taken at their extreme consequences, cannot be satisfied by any agent ultimately preventing communication [1.42]. The last problem is that these long conversations are often constructed by agents that do not share the same goal, and that may very easily lose track of the coherence of the conversation. Furthermore, the reasoning required to maintain coherence of the conversation is extremely difficult to model, as, for example, it requires employing abductive reasoning, to construct a model of the other parties involved in the conversation to "guess" their goals [1.23, 1.27]; but abductive reasoning is rarely adopted by software agents

In addition to the problems listed above, agents often enter in very scripted interactions where the meaning of speech acts is artificially modified to fit the situation. For instance, e-commerce agents enter auctions and bid to buy widgets there. Those agents need to follow the protocol of the auction. Furthermore, e-commerce agents need to understand what counts as a valid bid, how much commitment is associated with the bid, what is the cost of retracting the bid. Much of this information cannot be represented using speech acts. While there has been a great deal of work on agents in e-commerce on exploiting game theoretic reasoning to select a winning auction strategy [1.40], from the communication point of view, it is always assumed that the agent is able to participate in the auction.

Agents interactions are very constrained and goal directed. Agents do not entertain each other with small talk, rather their interactions are often very well defined and constrained by the context. In the example of e-commerce agents whose interaction is constrained by the auction protocol, any communicative action that is outside what is prescribed by the auction protocol is inappropriate and potentially dangerous when interpreted as a bid. From this point of view, the power of speech acts only adds non-needed complexity to long conversations. In general, each conversation is regulated by *Conversational Policies* , or in other words "set of principles that constrains the nature and exchange of semantically coherent ACL messages" [1.20].

Ontologies provide a way to formalize conversational policies as objects in the domain so that they can be used by the agents to participate in long conversations. For example, an auction may be formalized as an object that has a protocol, a number of participants that should have well defined char-

acteristics and that play well defined roles in the auction. Furthermore, an auction is characterized by negotiation rules, products to sell, and bids placed by the potential buyers. The use of ontologies to formalize conversations has two advantages: first of all it specifies in a precise and declarative way what agents should do to be part of the conversations, or as in our example of the auction, the second advantage is that the same knowledge is shared across all the agents so that they not only know what they have to do but they also have expectations on what the auctioneer and the other bidders plan to do.

A similar approach is followed by DAML-S [1.2] when it defines the interaction protocol with an agent[6]. Using DAML-S the agents specifies the protocol of interaction with others as a workflow that mimics its own processing workflow. Furthermore it specifies what information it needs as input what it generates as output and the format of the messages exchanged. Using DAML-S allows any agent to infer the interaction protocol and to decide what information to send at any given time.

The use of ontologies to represent conversations is producing a shift in the interpretation of agent communication languages. By representing explicitly conversations as protocols and the set of messages that have to be exchanged, the ontological representation effectively departs from the traditional representation of agent communication based on speech acts. Indeed, when the protocol is completely specified there is no need to use speech acts at all. The cost of this switch is the transition from an extremely powerful way to represent agent messages based on speech acts where agents have the ability to say all they want at any time, to a constrained way that carefully specifies the interaction protocol limiting the messages that can be exchanged with the advantage of gaining management of the conversation.

1.4 Social Knowledge

Communication achieves two objectives for agents. First it allows agents to work together in such a way that the work of one agent contributes to the work of other agents. For example, the performance of the requester of information, depends on the quality of the information provided and the timely answers of the provider. The second contribution of communication is to provide a coordination mechanism whereby agents negotiate how they are going to share a common resource or how they are going to collaborate toward the solution of a problem.

Agent's interaction and collaboration results in the emergence of a society of agents with its own (often implicit) social structure. As a consequence agents need *social knowledge* [1.25] which specifies how agents work together

[6] DAML-S is designed to represent "Semantic" Web services, which are, from our prospective, indistinguishable from agents.

and the *norms* of acceptable social behavior in the MAS [1.5, 1.9]. For example, asking an agent to perform an action or to provide information implies a commitment on the part of the speaker of waiting for the answer, and a commitment on the part of the listener to respond truthfully, coherently, and in a timely manner to the request [1.42]. When agents do not live up to their commitments, the interaction fails, therefore either the listener fails to receive an appropriate answer, or the speaker will not be able to deliver its answer to the listener. Norms provide a tool to represent explicitly what is expected by an agent in a given situation; the utility of norms is in making the MAS and the agents in it more predictable. Because of the existence of norms, agents can develop predictions on the behavior of other agents, which in turns allow higher levels of cooperation.

At the knowledge level, norms specify constraints on the behavior of the agent within what is acceptable for the rest of the agents at large. In general, it is up to the agent to decide whether to adhere or violate a norm, knowing that violations of the norm come at some cost for the agent itself, or for other agents in the MAS. Violations may result when the agent believes that the adherence to the norm is either too costly or it prevents the agent from the achievement of its goals.

While in principle, norms can be used to express any social constraint, they find their immediate application in the definition of concepts like contracts and commitments that agents develop as they work together. For example, contracts carry the obligation of the parties to perform their role in the contract, and the authorization to take some action when the contract is violated [1.11]. For example, the obligation of an airline to transport all passengers that have a reservation and the obligation of the passengers of paying for their ticket are shown in figure 1.9. Here the `goal` condition represents the goal that the agents have to fulfill their obligation, while the `exit` condition represents the condition that results if the agents do not fulfill their obligations.

```
obligation(airline,passenger,transport(airline,passenger)
   in:   flightReservation(passenger,airline)
   goal: transport(passenger,airline)
   exit: cancel(passenger,ticket) ⟹
                obligation(passenger,airline,payCost(passenger,airline)) ∧
                auth(airline,passenger,direct(payCost(passenger,airline)))
         cancel(airline,flight) ⟹
                obligation(airline,passenger,payCost(airline,passenger)) ∧
                auth(passenger,airline,direct(payCost(airline,passenger)))
```

Fig. 1.9. Example of an obligation

Authorizations provide a permission to the agent to perform an action. For instance, our example of obligation shown in figure 1.9 specifies that the passenger is authorized to ask for a reimbursement if the flight is canceled. This authorization allows the passenger to actually ask for the reimbursement. The formal definition of this authorization is shown in figure 1.10. The authorization allows the passenger to assume the goal of making the airline pay the cost of canceling the reservation.

```
auth(passenger,airline,direct(payCost(airline,passenger))
   in:   flightReservation(passenger,airline)
   goal: payCost(airline,passenger)
```

Fig. 1.10. Example of an authorization

Norms provide ontologies for the specification of the expected social behavior in a given community of agents. To this extent, ontologies provide the language to express norms that are shared by every agent in the MAS; furthermore, ontologies provide the conceptual structures to represent norms as shared concepts. Despite the importance of expressing norms in the agent community, virtually no work has been done to develop an ontological representation of norms. Much of the contribution of this community has been in the study of the effects of norms on the problem solving process of the agent.

In addition to the work on norms in the agent community, there are other two sources of ontologies to represent concepts similar to norms. The first contribution is the representation of laws in legal reasoning. [1.53, 1.52]. The main task of these ontologies is indexing of laws and regulations for information retrieval and possibly for validity checking to verify that the regulations do not contradict each other. These ontologies provide a different notion of norm which is basically a description of a law, and the acts to which those norms apply, as for instance robbery or murder.

The second effort to construct ontologies for norms comes from the e-commerce community and their needs to express concepts such as contracts that regulate electronic transactions. SweetDeal [1.22] bases its representation on a combination of Semantic Web ontology languages such as DAML and RuleML [1.37] in conjunction with the MIT Process Handbook. Contracts are associated with a process such as selling, or delivery and with a set of exceptions which describe what can go wrong, such as late delivery, with the contract and the associated penalties. Furthermore, this work shows how contracts can be used in conjunction with business rules to manage risk of penalties

1.5 Open Challenges

Despite the considerable effort toward the construction of ontologies that allow agents to interoperate in open MAS, very difficult and challenging problems are still open. We can divide these problems in two major sets of challenges: the first set of challenges are related to the type of languages and logics that provide sufficient expressive power to be of use for agents in their interactions; the second set of challenges lies in the creation of ontologies that help the agents in their everyday interaction in MAS. We will leave the first set of challenges to other chapters of this book, and we will concentrate on the second set of challenges.

A major challenge comes with the description of ontologies that describe concepts such as commitments, agreements and contracts. As shown in [1.21] these ontologies prove to be essential for a wide spread use of the Semantic Web and Semantic Web services [1.2] in real World e-commerce applications. Crucially, there seems to be a schism between the use and representation of norms that comes from the agent community which is mostly concerned with the mental attitudes that the agent adopts as a consequences of accepting a given norm; and the Semantic Web attitude of providing a description of contracts that is consistent with the underlying semantics of Semantic Web languages leaving to the proof theory of the language and the inference engines adopted by the agents the derivation of the appropriate consequences.

In addition to the development of ontologies for commitments, agents and Semantic Web services need ways to express and reason about the reputation and trust of the agents that they encounter. These are fundamental evaluation criteria that we typically use when we enter in a business relation with our partners. These concepts need to be expanded to communities of agents so that a cheating agent is penalized by its unacceptable behavior. Some seminal work in the area is represented in [1.4, 1.55, 1.56].

Agents can interoperate if they adopt the same modes of communication, including agreeing on the security protocols used in their communication. Ontologies are needed to specify the security requirements of agents [1.10].

Throughout the paper we referred generically to "ontologies" as if there is a consistent corpus of ontologies available to the agents. In reality many ontologies are redundant in the sense that they present the same domain, but with little or no interoperation between the ontologies [1.44]. Since ontologies play a pivotal role in agent interoperation by providing a shared representation of the domain and of the concepts that the agents need to use, agents that use different ontologies fail to interoperate. Effectively, it is as if the two agents end up speaking two different languages. The problem of ontology interoperation is very difficult since ontologies may provide different prospective and different sets of information on the concepts that they represent.

10

1.1 Bernstein Abraham and Mark Klein. High precision service retrieval. In *ISWC 2002*, Sardegna, Italy, 2002.
1.2 The DAML Services Coalition: Anupriya Ankolekar, Mark Burstein, Jerry R. Hobbs, Ora Lassila, David L. Martin, Sheila A. McIlraith, Srini Narayanan, Massimo Paolucci, Terry Payne, Katia Sycara, and Honglei Zeng. Daml-s: Web service description for the semantic web. In *ISWC 2002*, Sardegna, Italy, 2002.
1.3 J.L. Austin. *How to Do Things with Words 2nd Edition.* Harvard University Press, Cambridge, Ma., 1962.
1.4 Jose Cardoso, John Miller, Amit Sheth, and Jonathan Arnold. Modeling quality of service for workflows and web service processes. *Very Large Data Bases Journal*, 2002.
1.5 Cristiano Castelfranchi, Frank Dignum, Catholijn M. Jonker, and Jan Treur. Deliberate normative agents: Principles and architecture. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI.* Springer-Verlag, Berlin, 2000.
1.6 Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Description Language (WSDL) 1.1. http://www.w3.org/TR/2001/NOTE-wsdl-20010315, 2001.
1.7 Philip R. Cohen and C. Raymond Perrault. Elements of a plan-based theory of speech acts. *Cognitive Science*, 3:177–212, 1979.
1.8 DAML Joint Committee. DAML+OIL language, 2001.
1.9 Rosaria Conte, Cristiano Castelfranchi, and Frank Dignum. Autonomous norm acceptance. In Jörg Müller, Munindar P. Singh, and Anand S. Rao, editors, *Proceedings of the 5th International Workshop on Intelligent Agents V : Agent Theories, Architectures, and Languages (ATAL-98)*, volume 1555, pages 99–112. Springer-Verlag: Heidelberg, Germany, 1999.
1.10 Grit Denker. Towards security in daml. Technical Report 94025, SRI International, 2002.
1.11 Frank Dignum. Autonomous agents and social norms. In *Workshop on Norms, Obligations and Conventions (ICMAS-96)*, Kyoto, 1996.
1.12 Kutluhan Erol, James Hendler, and Dana S. Nau. Htn planning: Complexity and expressivity. In *AAAI94*, Seattle, 1994.
1.13 Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 5:189–208, 1971.
1.14 Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In Jeff Bradshaw, editor, *Software Agents.* MIT Press, Cambridge, 1997.
1.15 FIPA. FIPA ACL message structure specification, 2002.
1.16 Michael P. Georgeff and Amy L. Lansky. Reactive reasoning and planning. In *AAAI87*, pages 677–682, Seattle, WA, 1987.
1.17 Malik Ghallab et. al. Pddl-the planning domain definition language v. 2. Technical Report, report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.
1.18 Giuseppe De Giacomo, Yves Lesprance, and Hector Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artificial Intelligence*, 121((1-2)):109–169, 1995.
1.19 Yolanda Gil and Surya Ramachandran. Phosphorus: A task-based agent matchmaker. In *Agents 2001*, Montreal, Canada, 2001.
1.20 Mark Greaves, Heather Holmback, and Jeffrey Bradshaw. What is a conversation policy? In Frank Dignum and Mark Greaves, editors, *Issues in Agent Communication*, pages 118–131. Springer-Verlag: Heidelberg, Germany, 2000.

1.21 Benjamin N. Grosof. Representing e-business rules for the semantic web: Situated courteous logic programs in ruleml. In *WITS '01*, 2001.
1.22 Benjamin N. Grosof and Terrence C. Poon. Representing agent contracts with exceptions using xml rules, ontologies, and process descriptions. In *International Workshop on Rule Markup Languages for Business Rules on the Semantic Web*, 2002.
1.23 Jerry R. Hobbs, Mark Stickel, Paul Martin, and Douglas Edwards. Interpretation as abduction. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, NY, 1988.
1.24 Marcus J. Huber. Jam: A bdi-theoretic mobile agent architecture. In *Agents'99*, Seattle, WA, 1999.
1.25 N. R. Jennings and J. R. Campos. Towards a social level characterisation of socially responsible agents. *IEE Proceedings on Software Engineering*, 144(1):11–25, 1997.
1.26 Larson K. and T Sandholm. Computationally limited agents in auctions. In *Workshop on Agent-based Approaches to B2B*, 2001.
1.27 Henry A. Kautz and James F. Allen. Generalized plan recognition. In *AAAI-86*, Philadelphia, Pa., 1986.
1.28 Y. Lespérance, H. Levesque, F. Lin, and R. Scherl. Ability and knowing how in the situation calculus. *Studia Logica*, 66(1):165–186, October 2000.
1.29 David Lewis. Scorekeeping in a language game. *Journal of Philosophical Logic*, 1979(8):339–359, 1979.
1.30 Thomas W. Malone, Kevin Crowston, Brian Pentland Jintae Lee, Chrysanthos Dellarocas, George Wyner, John Quimby, Charles S. Osborn, Abraham Bernstein, George Herman, Mark Klein, and Elissa O'Donnell. Tools for inventing organizations: Toward a handbook of organizational processes. *Management Science*, 45(3):425–443, March, 199 1997.
1.31 David Martin, Adam Cheyer, and Douglas Moran. The Open Agent Architecture: A Framework for Building Distributed Software Systems. *Applied Artificial Intelligence*, 13(1-2):92–128, 1999.
1.32 Jörg P. Müller. *The Design of Intelligent Agents*. Springer, 1996.
1.33 Marian Nodine, Jerry Fowler, Tomasz Ksiezyk, Brad Perry, Malcolm Taylor, and Amy Unruh. Active information gathering in infosleuth. *International Journal of Cooperative Information Systems*, 9(1/2):3–28, 2000.
1.34 M. Paolucci, D. Kalp, A. Pannu, O. Shehory, and K. Sycara. A planning component for RETSINA agents. In N.R. Jennings and Y. Lespérance, editors, *Intelligent Agents VI*, Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin, 2000.
1.35 Massimo Paolucci, Takahiro Kawamura, Terry R. Payne, and Katia Sycara. Semantic matching of web services capabilities. In *ISWC2002*, 2002.
1.36 Anand S. Rao and Michael P. Georgeff. Modelling rational agents within a bdi-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning*, Cambridge, MA, 1991.
1.37 RuleML. The RuleML Web Site. http://www.dfki.uni-kl.de/ruleml/.
1.38 G. Salton and A Wong. A Vector Space Model for Automatic Indexing. *Communications of ACM*, 18:613–620, 1975.
1.39 Tuomas Sandholm. Distributed rational decision making. In G. Wei, editor, *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence*, pages 201–258. MIT Press, 1999.
1.40 Tuomas Sandholm. emediator: A next generation electronic commerce server. *Computational Intelligence*, 18(4):656–676, 2002.
1.41 John R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, 1969.

1.42 Munindar P. Singh. Agent communication languages: Rethinking the principles. *IEEE Computer*, 31(12), 1998.
1.43 R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, 29(12):1104–1113, 1980.
1.44 Gerd Stumme and Alexander Maedche. Ontology merging for federated ontologies on the semantic web using fca-merge. In *Workshop on Ontologies and Information Sharing (IJCAI '01)*, 2001.
1.45 Bill Swartout, Yolanda Gil, and Andre Valente. Representing capabilities of problem-solving methods. In *IJCAI 99 Workshop on Ontologies and Problem-Solving Methods.*, 1999.
1.46 Katia Sycara, Keith Decker, Anadeep Pannu, Mike Williamson, and Dajun Zeng. Distributed intelligent agents. *IEEE-Expert*, 11(6):36–45, 1996.
1.47 Katia Sycara and Mattheus Klusch. Brokering and matchmaking for coordination of agent societies: A survey. In Omicini et al, editor, *Coordination of Internet Agents*. Springer, 2001.
1.48 Katia Sycara, Mattheus Klusch, Seth Widoff, and Janguo Lu. Dynamic service matchmaking among agents in open information environments. *ACM SIGMOD Record (Special Issue on Semantic Interoperability in Global Information Systems)*, 28(1):47–53, 1999.
1.49 Katia Sycara, Massimo Paolucci, Martin van Velsen, and Joseph Giampapa. The RETSINA MAS infrastructure. *Autonomous Agents and Multiagent Systems*, 2003. forthcoming.
1.50 David Trastour, Claudio Bartolini, and Javier Gonzalez-Castillo. A semantic web approach to service description for matchmaking of services . In *SWWS2001*, Stanford, Ca, 2001.
1.51 UDDI. The UDDI Technical White Paper. http://www.uddi.org/, 2000.
1.52 Pepijn Visser and Trevor Bench-Capon. The formal specification of a legal ontology. In *Legal Knowledge Based Systems; foundations of legal knowledge systems. Proceedings JURIX'96*, 1996.
1.53 R. Winkels, A. Boer, and R. Hoekstra. Clime: Lessons learned in legal information serving. In *ECAI2002*, 2002.
1.54 Hao-Chi Wong and Katia Sycara. A Taxonomy of Middle-agents for the Internet. In *ICMAS'2000*, 2000.
1.55 Bin Yu and Munindar P. Singh. An evidential model of distributed reputation management. In *AAMAS2002*, Bologna, 2002.
1.56 Giorgos Zacharia, Alexandros Moukas, and Pattie Maes. Collaborative reputation mechanisms in electronic marketplaces. In *HICSS*, 1999.