Available online at www.sciencedirect.com
SCIENCE DIRECT®
Engineering Applications of Artificial Intelligence 18 (2005) 223–236
Engineering Applications of ARTIFICIAL INTELLIGENCE
www.elsevier.com/locate/engappai

Ontologies for supporting negotiation in e-commerce

Valentina Tamma[a,*], Steve Phelps[a], Ian Dickinson[b], Michael Wooldridge[a]

[a]Department of Computer Science, University of Liverpool, Peach Street, Liverpool L69 7ZF, UK
[b]Hewlett-Packard Research Labs, Filton Road, Bristol BS34 8QZ, UK

Received 5 November 2004

Abstract

In this paper we present our experience in applying Semantic Web technology to automated negotiation. This result is a novel approach to automated negotiation, that is particularly suitable to open environments such as the Internet. In this approach, agents can negotiate in any type of marketplace regardless of the negotiation mechanism in use. In order to support a wide variety of negotiation mechanisms, protocols are not hard-coded in the agents participating to negotiations, but are expressed in terms of a shared ontology, thus making this approach particularly suitable for applications such as electronic commerce. The paper describes a novel approach to negotiation, where the negotiation protocol does not need to be hard-coded in agents, but it is represented by an ontology: an explicit and declarative representation of the negotiation protocol. In this approach, agents need very little prior knowledge of the protocol, and acquire this knowledge directly from the marketplace. The ontology is also used to tune agents' strategies to the specific protocol used. The paper presents this novel approach and describes the experience gained in implementing the ontology and the learning mechanism to tune the strategy.
© 2004 Elsevier Ltd. All rights reserved.

Keywords: Ontologies; Agent negotiation; Semantic web technology

1. Introduction

Interest in automated negotiation in multiagent systems has been stimulated to a great extent by the vision of software agents negotiating with other software agents to buy and sell goods and services on behalf of their owners in a future Internet-based global marketplace. Broadly, negotiation can be understood as the process of reaching agreement on one or more matters of common interest.

Until now, research has focused on accounting for particular interactions among agents by developing and improving specifically tailored *negotiation protocols* and *strategies* (Rosenschein and Zlotkin, 1994; Kraus, 1997, 2001; Jennings et al., 2001); where the former refers to the rules the agent has to follow in order to participate to a negotiation (also known as *rules of encounter*), while the latter refers to the rationale for choosing among different actions at a certain stage (Rosenschein and Zlotkin, 1994).

Traditional negotiation approaches pose a number of constraints on the type of interactions that can take place among agents, for example only agents identified in advance or controlled can participate to a negotiation, and only pre-determined protocols are allowed. But, most importantly, usually protocols are coded *implicitly* within agents, as part of their code, and any modification to the protocol implies that the agents are taken off line in order to be reprogrammed.

However, e-commerce applications are becoming increasingly popular, and they are characterised by flexible, dynamic scenarios, where agents crawl the internet in search of a suitable marketplace for selling or buying, and the interaction rules can change within an

*Corresponding author. Tel.: +44 151 794 6797; fax: +44 151 794 3715.
E-mail addresses: valli@csc.liv.ac.uk (V. Tamma), sphelps@csc.liv.ac.uk (S. Phelps), ian.dickinson@hp.com (I. Dickinson), mjw@csc.liv.ac.uk (M. Wooldridge).

0952-1976/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.engappai.2004.11.011

interaction or between interactions. Agents are free to enter these marketplaces and engage in negotiation; the only pre-requisite is that agents will share some background knowledge and commit to some common rules of encounter.

One of the reasons behind the growth in popularity of e-commerce application is the availability of new technologies that rely upon the explicit representation of knowledge by means of machine interpretable languages such as RDF(S) (Decker et al., 2000) and XML (2001). Along these now well-established standards, new emerging paradigms such as the Semantic Web (Berners-Lee, 1999; Fensel et al., 2001; Hendler, 2001) are building a new, dynamic level on the Internet, offering markup languages with richer expressive power, such as DAML+OIL (DAML, 2001) and its successor OWL, which allow the creation of *ontologies* and their instantiation in the description of specific Web sites. These markup languages become the building blocks for providing representations of both static information, such as the quantity of goods and their price, but also of actions or change the state of the world, such as the actual sale of a product and business rules that regulate these actions, for instance the discount to be applied on sales verifying some constraints.

These new paradigms and the growing interest in e-commerce make it possible to conceive less restrictive applications that operate in open environments, where the problem of automated negotiation needs to be addressed with by designing approaches where fewer limitations are imposed on the agents and on the types of interaction they can be involved in. This paper describes a novel approach to automated negotiation based on the declarative and explicit representation of the negotiation mechanism, and which is therefore particularly suited to open environments. This approach utilises *ontologies* (Studer et al., 1998) to make the representation of the rules of encounter *explicit*, *machine readable* and *sharable*; agents willing to participate to a negotiation session commit to the shared ontology, which explains them how the mechanism governing the negotiation works. The ontology is also used as input to a learning algorithm that is used by the agents to tailor their strategy to the specific negotiation protocol implemented in the marketplace. The approach was developed as part of a project funded by HP Labs that aimed to investigate ontology-based representation of agent capabilities.

2. Ontology-based approach to automated negotiation

Automated negotiation is typically based on the assumption that agents can only participate to a negotiation if they commit to a shared protocol. In most traditional negotiation scenarios, the protocol is fixed and implicitly assumed: an agent that engages in negotiation is assumed to know and agree to the protocol a priori. The agent is also assumed to be equipped with a strategy that permits it to participate to the negotiation with the goal of maximising its welfare (Kraus, 2001).

This type of negotiation is suitable to closed or semi-open negotiation environments, where the agents taking part in the interaction have been identified in advance or are controlled, and only pre-determined protocols are allowed. But advances in technologies are pushing toward increasingly open negotiation environments, where agents are free to enter and participate; the only pre-requisite required is that agents will share some background knowledge and commit to some common rules of encounter. These environments are flexible and dynamic, the interaction rules can change within an interaction or in between interactions. In these environments assuming a fixed and immutable negotiation protocol is no longer viable.

In this paper, we propose an approach to automated negotiation that fully exploits the potential of open environments: agents should not be forced to commit to a single negotiation protocol, *but should be able to choose the negotiation protocol which is most suitable to the type of interaction they participate in*. This, in turns, implies also that agents should be able to tune their strategy to the specific protocol employed, in order to maximise their chances of success in the negotiation.

In this approach, negotiation protocols are not hard-coded in agents, but instead when a new agent joins a pre-existing interaction, it receives an advertisement communicating the type of protocol regulating the interaction and describing it in terms of a *shared ontology* of negotiation. The approach proposed in this paper is novel in that no other approach presented in the literature makes use of a shared ontology of negotiation in order to model negotiation protocols. The approach is more similar to those proposed for sharing knowledge among heterogeneous resources (Neches et al., 1991): agents inter-operation is enabled via a shared ontology which provides a formal and agreed upon definition of the terms that are to be used by the agents.

In the negotiation context, the shared ontology provides the basic vocabulary that an agent and a *negotiation host* (who is responsible for the creation and enforcement of the rules governing participation (Bartolini et al., 2002)) must share in order to discuss the terms of the participation to the negotiation session, together with a declarative representation of the rules that describe the conditions under which the interaction between agents takes place, the deals that can be made, and the permitted sequences of offers (Lomuscio et al., 2000). Therefore, agents interaction is no longer regulated by a specific negotiation protocol hard-coded within the agent, but the shared description of the

protocol is now "acquired" from the marketplace where the inter-operation takes place. The negotiation ontology is also used to "tune" the agent strategy to the specific rules of encounters of the negotiation session. Since the strategy is private to the agent and cannot be shared, the tuning is performed by a machine learning approach, which permits the agent to adapt its strategy by learning the value of some parameters that determine the course of action of the agent.

This means that agents no longer need to go offline and be reprogrammed in order to participate to other kinds of negotiations, thus overcoming a limitation of traditional negotiation approaches when applied to open environments, such as those enabling electronic trading and the Semantic Web, which require a flexible type of interaction, in that agents should support a wide variety of negotiation forms.

This ontology acts as a general framework that permits agents to reach agreement. Using a shared ontology of protocols makes it easier to compare the different negotiation protocols and to understand similarities and differences, thus facilitating agreement on a single protocol. The most general concepts (shared across all possible applications and domains) in the ontology are represented in the upper part of the hierarchy. By refining the concepts which compose the ontology we describe groups of similar protocols, thus we define the kind of features that are common to all of them. Concepts in the lowest part of the hierarchy are quite specific, and concern a single negotiation protocol. The refinement of a concept is obtained by restricting the values associated with the attributes describing the concepts, or by adding new attributes which enrich concept descriptions. Applicability rules and constraints are represented by axioms.

The advantages of this approach are of twofold. The first advantage is flexibility. Negotiation protocols but can be learned dynamically by acquiring the part of ontology modelling them. The second type of advantage is that the ontology provides the terminology to reason in terms of negotiation protocols, their components, and the constraints regulating them. The term definitions can be used as a classification framework that permits the analysis of the negotiation protocols available, and to develop new ones. Moreover, the commitment to the same high level concepts can facilitate communication of negotiation rules among agents, thus improving flexibility.

3. Sharing knowledge about the negotiation domain

The decisions regarding the concepts to be included in an ontology depend on a careful analysis of the domain to be modelled and the purpose for which the ontology is built. In our approach, the ontology serves the main purpose of facilitating automated negotiation, which in turns, can be decomposed into the following subtasks:

(1) it allows agents to engage in negotiation without prior knowledge of the negotiation mechanism used;
(2) it enables agents to exchange knowledge about arbitrary negotiation mechanisms;
(3) it reduces the amount of knowledge hardcoded in the agent;
(4) it facilitates commitment to a *shared view* of the negotiation domain, where agents agree on the meaning of the concepts used in describing the negotiation process.

However, merely committing to the shared ontology for negotiation does not imply that agents are able to choose the *best course of action*, that is, the actions that make the agents maximise their utility function. In order to choose the best course of action, agents need the ability to develop an appropriate *strategy* at run time. Strategy has no great impact in designing an ontology for negotiation, it is usually determined by means of a utility function calculated from some parameters, (such as price), which are also used to define the rules of encounter. In addition, the strategy is usually private to an agent (or the same agents belonging to a same user/organisation) and cannot and should not be shared among those involved in the negotiation. For this reason, the shared ontology for negotiation models a shared view of the entities involved in a negotiation process, as well as the states and transitions determined by the rules of encounter. Few concepts depend on the strategy, and the ontology serves primarily as input to a learning process that it is used to tune the generic strategy to the specific type of protocol.

Building on the assumptions made and on the role the ontology has to serve, we can identify the following modelling requirements, which guided us in the design of the shared ontology for negotiation:

(1) The ontology provides a description of the negotiation domain, by identifying the objects which are relevant for the domain and the relationships connecting them, for instance the ontology defines concepts such as negotiation protocol, offer, bid, auction, etc., relationships between them, such as a protocol is composed of a number of offers greater than zero, or constraints such as a winning offer is the one with the highest price;
(2) The ontology also models the negotiation *process* itself, that is the states and the state transitions defining the process;
(3) The ontology is a commitment to a shared view, therefore it needs to be shared and to represent consensus on the objects of the domain.

In order to express consensus, the negotiaton ontology we have developed builds on previous efforts to find commonalities across different negotiation protocols. From an analysis of the classification framework illustrated in Lomuscio et al. (2000), the generic software framework for automated negotiation developed at HP Labs (Bartolini et al., 2002), the work by Wurman and colleagues (2001), and the London classification (Field et al., 2000), we have identified the concepts and the relationships that are shared across most negotiation protocols. These have been modelled as the higher level concepts in the negotiation ontology.

The lower level concepts in the negotiation ontology specify the roles played by the agents involved in a negotiation process and the rules that describe the stages and the features of a protocol, such as which agents are permitted to see the offers, how a negotiation terminates, etc. The rules we have considered in the ontology are those identified by Bartolini and colleagues in Bartolini et al. (2002), however, this set of rules is intended here only as an example, they are neither meant to be exhaustive nor have they been instantiated. They are intended to show a possible way of specialising the concept `Negotiation-rule` defined in the ontology. Fig. 1 illustrates a fragment of the negotiation ontology. We developed the ontology in Protegé 2000 (Fridman Noy et al., 2000), because of the expressive knowledge model, the possibility of dealing with axioms and the plugins that make it possible to translate the ontology to OWL, the Web Ontology Language that is the standard de facto for Semantic Web applications.

3.1. The ontology

One of the purposes of the shared ontology for negotiation is to facilitate sharing knowledge about an arbitrary protocol. For this reason, the ontology needs to model the notion of negotiation protocol, but also a number of relevant concepts, each of them highlighting a different aspect of a negotiation protocol:

- *Type of protocol*: This concept defines a generic protocol defining the "rules of encounter" that are followed by the negotiation participants during a negotiation process. The rules describe the conditions defining the interactions between agents, the deals that can be made and the permitted sequences of offers (Lomuscio et al., 2000).
- *Party*: This concept describes a single agent (be it human or electronic) or an organisation of agents which participate in a negotiation. Several agents can negotiate, and they can play different roles in the negotiation.

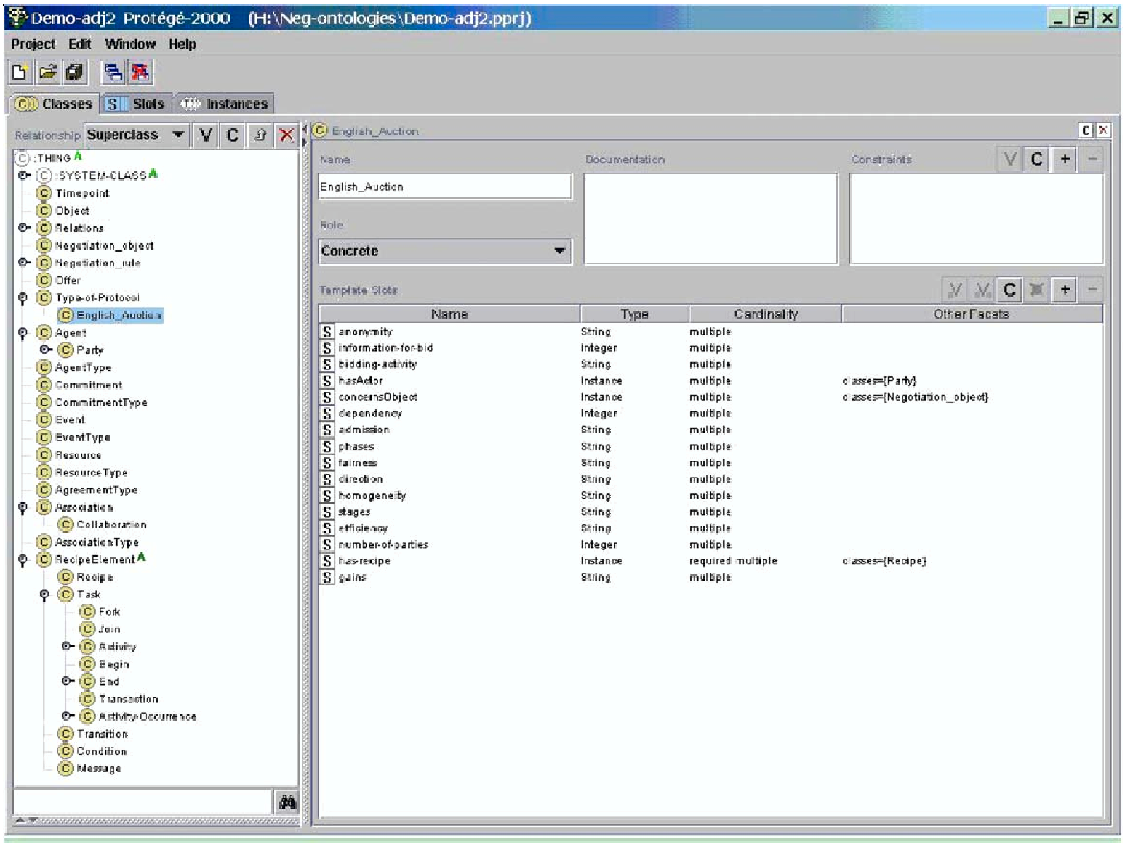

Fig. 1. Ontology screenshot from Protegé 2000.

- *Recipe*: We follow the terminology used in the resource, event and agent (REA) enterprise ontology (Haugen and McCarthy, 2000) to describe the process of reaching an agreement on some issues. The agreement can be obtained by modifying the values of the attributes of the negotiation. We chose part of the REA enterprise ontology because it is a well-established model, some of its concepts have been used to model the Business Requirements in UN/CEFACT modelling methodology (UMM, formally known as TMWG N090), and the Business Process Analysis Worksheets in ebXML.
- *Negotiation object*: It describes the objects of the negotiation, that is the material or immaterial goods that are transferred once an agreement has been reached.
- *Offer*: This concept describes a possible combination of values associated with the negotiation attributes which represents an expression of will (for example to purchase a certain number of goods, to receive the goods by a certain date, or to pay a maximum price for the goods).
- *Negotiation rule*: The set of rules that govern a specific negotiation protocol. The generic protocol is parametric with respect to the negotiation rules that are applicable to the type of electronic market modelled by the protocol. In the ontology this means that we identify a number of negotiation rules, and the way in which they are specified defines a specific negotiation protocol.

Ad hoc relationships between concepts are also defined, which describe how the identified concepts interact to define the negotiation protocol domain. For example, a `Type of Protocol` *Has recipe* `Recipe` which models the fact that a protocol is composed by a number of features, for instance the number of agents that can participate in negotiation, the phases, etc., but also by the mechanism itself that describes *how* the agents interact.

A protocol is also governed by a number of negotiation rules, and this aspect is modelled by the relationship (`Protocol` *Is governed by* `Negotiation-Rule`), where the concept `Negotiation-Rule` is specified by the different types of rules identified in Bartolini et al. (2002).

It should be noticed at this point that the higher level concepts of the negotiation ontology are not connected by an *IS A* relationship, since they are not taxonomic in nature, as already observed by Wurman and colleagues in Wurman et al. (2001). All the other concepts in the ontology are organised according to a proper *IS A* relationship (Lassila and McGuinness, 2001). The ontology is a heavy weight ontology where the concepts are constrained by a number of *axioms*, that need to be defined in order to be able to express the mechanism

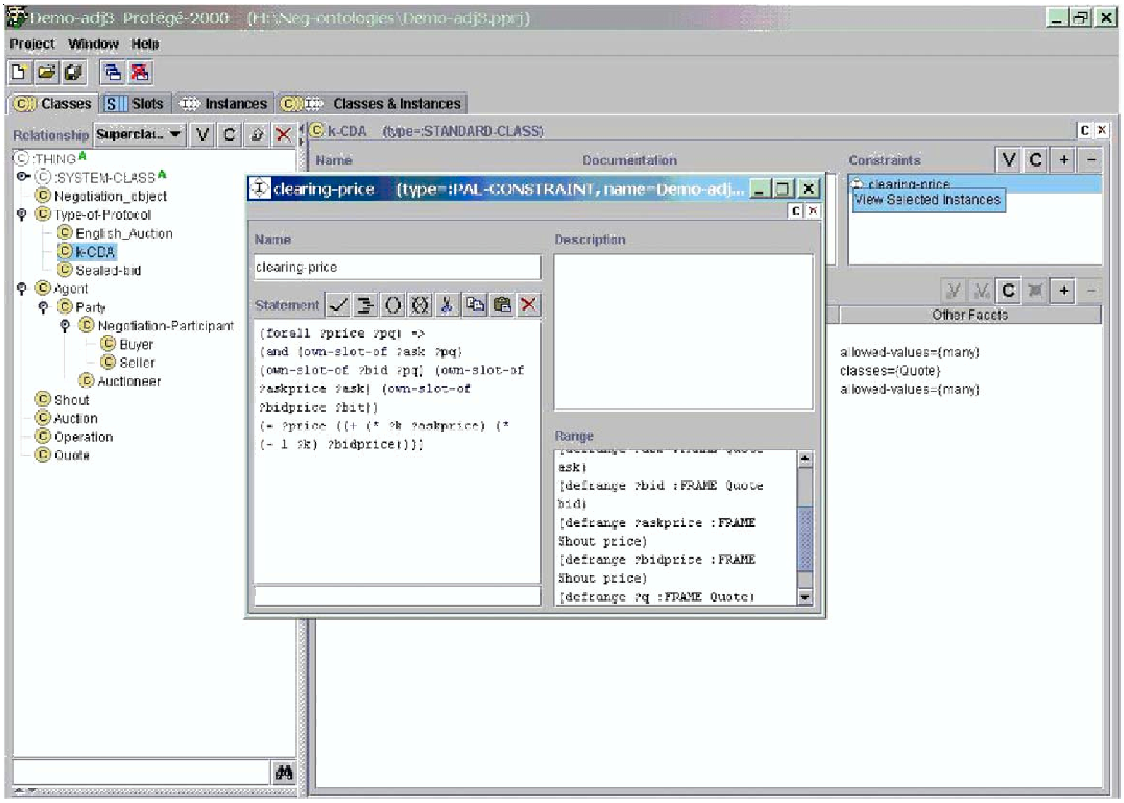

Fig. 2. An example of rule.

governing the protocol as a *process*. Fig. 2 illustrates the representation of axioms in the Protegé 2000 axiom language PAL.

4. Modelling the negotiation process

We view negotiation as a process whose transitions and states are described by the negotiation mechanism. From the ontology modelling viewpoint, this means that modelling domain factual knowledge, that is, knowledge concerning the objective realities in the domain of interest (Chandrasekaran et al., 1998) is not sufficient. We also need to model knowledge concerning the modality of interactions between agents, that is, the procedural flow of the protocol. Therefore, agents engaged in a negotiation process need to have two types of knowledge:

- Knowledge concerning the domain of interest: that is, the concepts which represent the objects that are negotiated and the parameters of the negotiation;
- Knowledge concerning the negotiation: that is, what are the roles involved in the negotiation and what are the permitted interactions.

This representation needs to be operative, in the sense that an agent equipped with the knowledge expressed in the negotiation ontology should be able to make use of the rules of encounter, either by executing them directly or to perform some sort of reasoning (for instance to check that the negotiation complies with the protocol used in the marketplace).

In Section 3.1 we have described the concepts modelling domain factual knowledge, here we concentrate on those concepts necessary to describe the procedural flow of the protocol. In order to make it executable, we took the design decision to view protocols as *processes*, and we base our modelling on the *Process Specification Language* (PSL).

The PSL defines a neutral representation for manufacturing processes. PSL has recently become a standard within Joint Working Group 8 of Sub-committee 4 (Industrial data) and Sub-committee 5 (Manufacturing integration) of Technical committee ISO TC 184 (Industrial automation systems and integration) (Gruninger, 2003).

PSL has the advantage of providing a formally defined semantics and allows for multiple syntaxes. The core of PSL are the formal definitions (ontology) that underlie the language. These formal and explicit representations permit us to share knowledge without having to rely on hidden assumptions or subjective mappings. PSL semantics are represented using Knowledge Interchange Format (KIF), a formal language developed as part of the Knowledge Sharing Initiative, in order to enable sharing knowledge among heterogeneous resources (Genesereth et al., 1992). KIF is a declarative language that can express arbitrary logical assertions and rules, thus through KIF expression, PSL can define concepts in a rigorous and unambiguous way.

The building blocks of PSL are KIF expressions defining the concepts modelled in the PSL ontology for processes. This ontology includes a set of terms and their definitions, and knowledge sharing is achieved through the commitment to share them. The interpretation of the KIF terms is constrained by the *KIF* axiomatisation. For example, one of the constraints on the KIF expression `(between a b c)` is described by the following axiom:

```
(defrelation between (?p ?q ?r) :=
(and (before ?p ?q) (before ?q ?r)))
```

saying that if we have three elements ?p, ?q, and ?r, and that ?p is before ?q and ?q is before ?r, then ?q is between ?p and ?r.

The PSL ontology consists of primitive terms, definitions, and the axiomatisation constraining their interpretation. The PSL ontology it is organised as layers of theories:

- *PSL Core*: The most basic elements of the PSL ontology.
- *Core theories*: Widely applicable extensions to PSL Core.
- *Extensions*: Definitions of process terminology for different applications, based on the Core.

The layers in the ontology allow for a modular organisation that facilitates the addition of future extensions and the support of multiple extensions that respond to a particular class of process specifications without having to support the whole PSL ontology.

In order to represent negotiation protocols as processes we relied mostly on the PSL Core, that specifies the concepts in the PSL ontology corresponding to the fundamental intuitions about activities. Therefore, we included in the ontology the key concepts of the PSL Core, such as:

- *Activity*: A class or type of action. For example, trading goods according to a English auction. It is the class of actions in which English auctions take place to trade goods.
- *Activity occurrence*: An event or action that takes place at a specific place and time, that is an *instance* or *occurrence* of an activity. For example, selling a green Micra in Liverpool, UK at 2 PM on July 2, 2004 is an occurrence of the "Trade goods with an English auction" activity.
- *Time point*: An instant separating two states, for example the point at which the Micra is offered

on the market, but before the first offer has been received.
- *Object*: Anything that is not a time point or an activity, for example the Micra.

In addition to the PSL concepts we also included some of the axioms, and we used them in order to offer a simplified way to represent processes and their state transitions.

5. Implementation example: the protocol mechanism

In order to allow agents to use the protocol as defined in the ontology, we need to translate the ontology into a suitable language that acts as a sort of *lingua franca* among the agents and enables the interoperation among them. The chosen language is DAML+OIL (Horrocks et al., 2002), [1] a markup language especially designed for enabling knowledge sharing among agents; its direct successor OWL is nowadays the standard de facto for Semantic Web applications.

The implementation of the approach outlined in Section 2 aims at verifying that the ontology can effectively support negotiation when agents have no prior knowledge of the protocol to be used in the marketplace. The ontology is used to advertise the rules of encounter characterising the mechanism, and to tune the agent strategy for the protocol used. As we will see in the remainder of the paper, we are still far from a scenario where agents need very little knowledge hardcoded about the protocol and are able to act on the basis of an explicit and machine processable description of the protocol mechanism. This is mainly due to the fact that, although expressive, OWL and its predecessor DAML+OIL, are not able to represent explicitly the semantics of the *rules* governing the protocol. In addition, there is a lack of tools that can be used to translate the axioms and rules into a suitable and executable language, for instance as rules in a rule base. However, we were more successful in using the ontology to tune the agent strategy through a learning mechanism described in Section 6.

In our approach, an agent committing to the ontology need only to create mappings between the ontology and its own internal view of the world (be it hardcoded in the agent or in a private ontology). Ideally, these mappings should be *translations* (Chalupsky, 2000), that is they should preserve the semantics of the concepts, however, *transformations* (Chalupsky, 2000) (mappings that do not preserve the semantics) are permitted. Transformations permit the relation of concepts in the ontology to the ones in the agent's conceptualisation which are most similar, typically a *hypernym* or a *hyponym*. In this way the agent's conceptualisation of the domain does not need to match completely the one of the negotiation ontology, but it has to be a "close approximation". By using this kind of approach, agents share only the concepts in the negotiation ontology, which are very general and make as few claims as possible about the world, thus respecting the minimal ontological commitment principle for knowledge sharing (Gruber, 1993).

The negotiation protocol is specific to the marketplace where the agent wants to interact; the *negotiation host* (that is, the agent that supervises the negotiation) advertises the URL of the ontology that is adopted to describe the negotiation protocol used in the marketplace, and in this example we assume to be the ontology presented in Section 3. The advertisement consists of the specification of the rules determining the protocol defined in terms of the negotiation ontology defined in Section 3, that is they are *strict subclasses of* the rules defined in the top level negotiation ontology. Let us suppose that the negotiation protocol adopted in the marketplace is an English auction, and that the negotiation object are tickets for entertainment events (such as concerts, or movies). The protocol, defined in DAML+OIL, is partially illustrated in Fig. 3 (we have not included the complete definition for reasons of space).

The concept `Protocol` in the ontology is defined in terms of other concepts such as `Object`, that is to be specified according to the specific rules of the protocol adopted, and the same happens to all the other concepts

```
<daml:Class rdf:about="#English-Auction">
  <rdfs:subClassOf rdf:resource="#Protocol"/>
  <rdfs:subClassOf>
    <daml:Restriction daml:minCardinality="3">
      <daml:onProperty rdf:resource:resource="#hasActor"/>
        <rdfs:comment>
          An English auction needs at least 2
          participants and 1 auctioneer
        </rdfs:comment>
      </daml:Restriction>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasObject"/>
      <toClass rdf:resource="#Protocol">
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>

<daml:Class rdf:ID="Object-English-Auction">
  <rdf:subClassOf rdf:resource="#Object"/>
  <rdf:subClassOf>
    <daml:Restriction daml:Cardinality="1">
      <daml:onProperty rdf:resource="#Attribute">
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

Fig. 3. Part of the English auction protocol definition.

[1] At the time of designing the ontology, the transition from DAML+OIL to OWL was still taking place, and the ontology was translated into DAML+OIL. All the examples in this paper are in DAML+OIL, but the ontology could be translated into OWL-Lite with little loss in the translation.

```
<daml:Class rdf:ID="Entertainment-Object-English-Auction">
  <rdf:subClassOf rdf:resource="#Object"/>
  <rdf:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#NumberOfItems"/>
      <daml:hasValue rdf:resource="#single"/>
    </daml:Restriction>
  </rdf:subClassOf>
</daml:class>

<daml:Class rdf:ID="Entertainment-Object-English-Auction">
  <rdf:subClassOf rdf:resource="#Object"/>
  <rdf:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#NumberOfItems"/>
      <daml:hasValue rdf:resource="#multiple"/>
    </daml:Restriction>
  </rdf:subClassOf>
</daml:class>
```

Fig. 4. The restrictions on the number of items negotiated.

which are related to `Protocol`. This means that the properties associated with these concepts will be filled with the values describing the restrictions posed on the protocol.

For example, in the case of an English auction for the entertainment domain the `NumberOfItems` is set to "Multiple" (in the hypothesis that multiple entertainment tickets are negotiated in the same negotiation process) whereas if we assume that only one ticket per auction can be negotiated, than `NumberOfItems` is set to "Single". The concepts are all specified by associating values with the properties of concepts, which is achieved by the restrictions illustrated in Fig. 4 and that should be advertised by the negotiation host.

Finally, the rules are specified. In order for an agent to be able to understand the rules, these should be defined in terms of the concepts modelled in the negotiation ontology. For instance, the negotiation ontology defines, among others, the rule describing the conditions under which an agreement can be made (*agreement formation rule*). Thus, the negotiation host advertises the agreement formation rule, which should be an instance of the one shown in Fig. 5.

Being an instance, all the "variables" (here represented by classes) should be instantiated in order for the rule to be applicable. The specific rule should be also expressed in a language which is executable. For example, we could represent rules in a rule engine such as the Java Expert System Shell (Jess). In this case, the ontology could be interpreted from DAML+OIL and fed into Jess to permit users to query the knowledge modelled in the ontology.

If we translate the ontology into Java and feed it to Jess, the agreement formation rule could be expressed as in Fig. 6 (Bartolini et al., 2002). The terms `BUYER` and `SELLER` should be defined in terms of the negotiation ontology, more precisely they should be defined as instances of the ontology concept `Party`, and both

```
<daml:Class rdf:ID=''Agreement-Formation-Rule-In-
                    English-Auction''>
  <rdfs:subClassOf
   rdf:resource=''#AgreementFormationRule''/>
/daml:Class>

<daml:ObjectProperty rdf:ID="hasRoleInput">
  <rdfs:domain rdf:resource="#Agreement-Formation-Rule-In-
                             English-Auction"/>
  <rdfs:range rdf:resource="#Party"/>
  <daml:minCardinality>2</daml:minCardinality>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="hasAttributeInput">
  <rdfs:domain rdf:resource="#Agreement-Formation-Rule-In-
                             English-Auction"/>
  <rdfs:range rdf:resource="#price"/>
  <daml:Cardinality>1</daml:Cardinality>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="hasRule">
  <rdfs:domain rdf:resource="#Agreement-Formation-Rule-In-
                             English-Auction"/>
  <rdfs:range rdf:resource="#Process"/>
  <daml:Cardinality>1</daml:Cardinality>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID="price">
  <rdf:type rdf:resource="#Object"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/
                            XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>
```

Fig. 5. The concept describing how to reach agreement in an English auction.

```
(defrule agreement-formation-rule)
(active-proposal}
    (proposal-id ?PID)
    (submitter ?BUYER)
    (role Buyer)
    (price ?PRICE))
(active-proposal
    (proposal-id ?PID)
    (submitter ?SELLER)
    (role Seller)
    (price ?RES-PRICE)
(test
    (> PRICE RES-PRICE))
=> (assert
   (agreeement}
      (buyer ?BUYER)
      (seller ?SELLER)
      (price ?PRICE)))
```

Fig. 6. The agreement formation rule expressed in Jess.

`RES-PRICE` and `PRICE` should be instances of the ontology concept `Price`.

The example of rule presented above shows that there is still a gap of expressive power between the static notions in the ontology, such as price, or agent, and

more dynamic notions, such as the restrictions posed on the protocol, that depend on the changes in the state of the world, and that are typically captured by means of axioms or rules. Static notions can be represented by languages such as OWL or DAML+OIL, whereas for dynamic notions we need to use rule engines, which permit agents to execute the rules. This implies that agents are equipped with the rule engine used to advertise the rules, and that the ontology concepts are translated into facts, to provide a semantic basis to the rules. This translation is not always straightforward, and needs reliable tools to support it, such as SweetJess (Grosof et al.,) and its corresponding tool for Prolog, SweetProlog (Laera et al., 2004). Unfortunately, these tools are not yet adequate to handle the complexity of rules governing negotiation protocols. Currently, there are a number of proposed markup standards for expressing business rules that constrain the definitions in the ontology, including RuleML (Boley et al., 2001) and OWL rules, and the effort to merge the two approaches SWRL (Horrocks et al., 2004), but most of them are still under development, although they are a step forward in the direction of representing dynamic state changes and workflows.

6. Implementation example: tuning the strategy

In the reminder of this section we describe how agents can use the ontology in order to determine an optimal negotiation strategy to adopt. In this approach we determine the agent strategy as a function of the rules governing the negotiation mechanism, therefore we use a declarative and executable form of the ontology obtained by translating it into a Jess rule base.

Determining the negotiation strategy to employ as a function of the mechanism rules turns out to be a very hard problem, and is not practically computable in the general case. For this reason, we use an approach that enables agents to acquire sufficiently good, but not necessarily optimal, strategies after several iterations of play against the mechanism devised and the other agents negotiating in the chosen marketplace.

In order to obtain a good deal in a negotiation mechanism, the agents need to bid *strategically*. Strategic bidding is a highly complex task: it involves reasoning about the negotiation mechanism and the strategies of other traders, and the other traders themselves are also reasoning about the agent strategy. Strategic reasoning is usually based on game theoretic techniques; unfortunately, determining the best strategy to play based on a description of the game is non-computable, apart from the most trivial cases. An alternative option, that we follow in our approach, is to equip our agents with simple reinforcement learning capabilities. These agents are not always able to discover the optimal strategy for a given mechanism, but the advantage is that they can acquire reasonably good strategies after several iterations of play against a new mechanism and the agents trading therein.

This type of approaches to acquire negotiation strategies are not new, however, approach differs in that we use an expert system, based on the negotiation ontology, to automatically select and configure the learning algorithms used.

6.1. The knowledge base describing the mechanism

We base our approach on that of Bartolini et al. (2002) and Wurman et al. (2001). In such frameworks, the job of the negotiation mechanism in its most abstract form is to take as input a set of offers to buy or sell from agents, and produce as output a set of potential transactions between agents, where a transaction is a tuple of the form (*buyer*, *seller*, *price*, *quantity*). The implementation of a specific negotiation mechanism needs to take into consideration a number of issues, that are represented in the ontology by axioms and rules, for example the algorithm to match offers to buy with offers to sell, or the mechanism to determine the transaction price depending on the offer prices. That is, whether the transaction price should be set on the highest bid, or on the second highest bid.

These issues depend heavily on the requirements of the negotiation mechanism. For example, sealed-bid auctions are more commonly used in private-value scenarios, where agents have a well-defined valuation for the resource that does not depend on other agents' valuations, whereas ascending auctions are used when valuations are interdependent. Therefore, the ontology (and its executable version in Jess) needs to model the entire space of the possible negotiation mechanisms. However, in this proof of concept we restrict the attention to the following aspects of the mechanism design space:

(1) the timing of the clearing operation, in which offers to buy are matched with offers to sell and transactions take place;
(2) the timing of the quote generation operation, in which information about the current state of the negotiation is issued to traders;
(3) the rule governing the transaction price;
(4) the visibility of agents' bids; and
(5) whether or not we allow multiple buyers or multiple sellers.

This means that we have not considered the whole ontology, but we have restricted our attention only to a subset of concepts and rules to be used in this proof of concepts. Figs. 7 and 8 show the concepts considered for the strategy implementation and an example of the rules

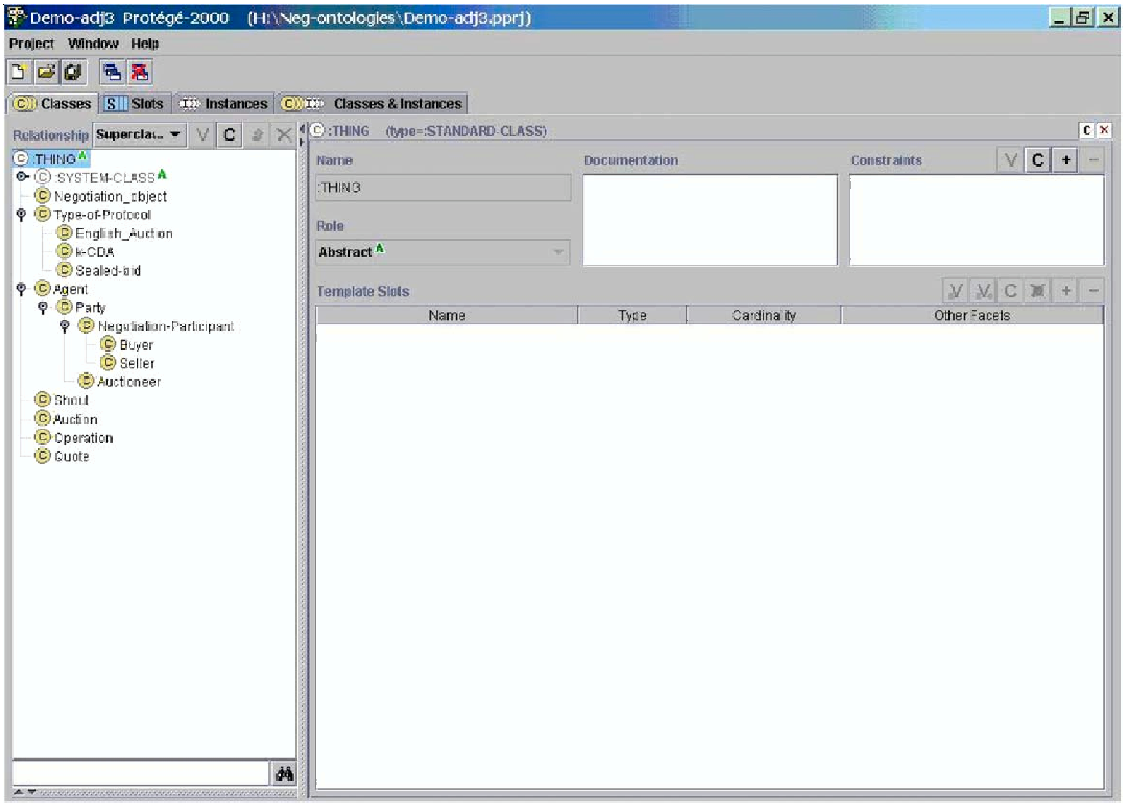

Fig. 7. The reduced ontology.

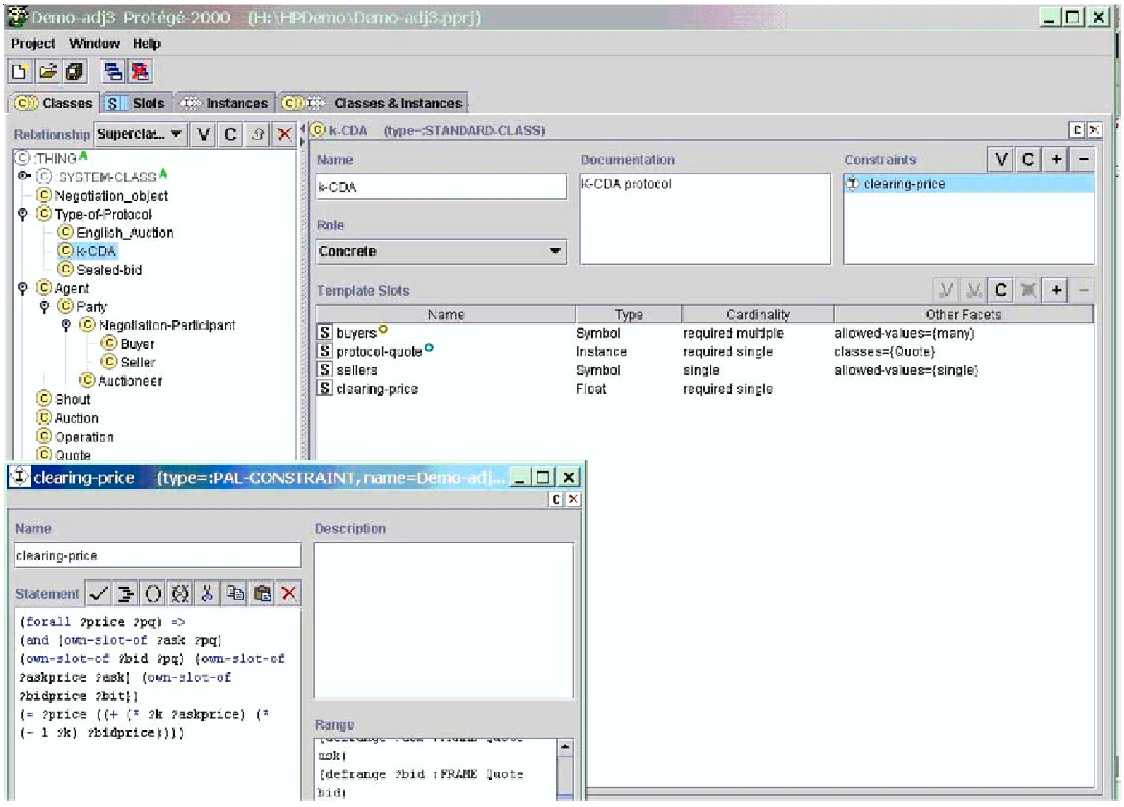

Fig. 8. The reduced ontology and the rules.

used in the implementation. These facts about the mechanism are then specified by Jess rules.

Each of the aspects above is governed by one or more Jess predicates. The main predicates are:

- *Clear timing and price generation*: In order to specify the timing of the operations, we define a predicate `auctioneer-operation-at` which specifies which operations are to be performed when a specified event occurs. In (Phelps et al., 2004) we give an exhaustive list of the possible events and actions. For example, in order to specify that clearing occurs continuously after every new offer is received, and that quotes are issued at the end of each round, we would assert the Jess facts:
  `(auctioneer-operation-at new-shout clear)`
  `(auctioneer-operation-at end-of-round generate-quote)`
- *Transaction price*: Transaction prices are typically a point in the interval between either: (i) the ask price and the bid price (*discriminatory price* mechanism); or (ii) the ask quote and the bid quote (*uniform price* mechanism).
  The transaction price is determined by means of a rule that asserts what price the agent should set the transaction at based on facts about the offers currently being cleared and the current market-quote. The aim of the transaction pricing rule is to assert a fact of the form:
  `(transaction-price ?price)`
  based on the facts about the current bid and ask and the current market-quote. The offers currently being cleared are represented as predicates of the form:
  `(clearask (?price) (?quantity))`
  `(clearbid (?price) (?quantity))`
  The current market-quote data is represented as a predicate of the form:
  `(quote (ask ?price) (bid ?price))`
  By convention, pricing rules are parameterised by a parameter k, which determines where in the interval between two prices we should set the transaction price. Typically, the price of a transaction is given by:
  $$p_t = kp_a + (1-k)p_b, \quad (1)$$
  where p_a is either the ask price or the ask quote and p_b is either the bid price or the bid quote, depending on whether we are using discriminatory or uniform pricing, respectively.
- *Bid visibility*: We constrain the offer visibility through the atomic predicate `shouts-visible`, which determines whether or not an agents' offers are visible to other agents.
- *Buyer to seller ratio*: The number of buyers allowed in the mechanism is a constraint posed on the number of buyers attribute of the negotiation protocol, and in our approach it is represented as a predicate of the form:
  `(buyers ?n)`
  Where `?n` can be `many` or `1`. The number of sellers in the mechanism is represented similarly as a predicate of the form:
  `(sellers ?n)`

The example in Fig. 9 describes a uniform-price k-CDA in Jess using our framework.

6.2. Implementing the approach to tune the strategies

Describing the declarative framework to use for specifying negotiation mechanisms is not sufficient to enable the agents to determine a good strategy, we also need some way of executing these specifications; that is, allowing an auctioneer agent to run an auction using the negotiation mechanism that is specified at run-time. To this end we utilise the JASA auction simulator (Phelps et al., 2004), a highly extensible open-source simulator designed for performing experiments in agent-based computational economics (Tesfatsion, 2001). This

```
(deffacts auctioneer-processing-facts

; We perform the clearing operation continuously;
;  every time a new shout arrives
(auctioneer-operation-at new-shout clear)

; Issue quotes at the end of each auction round
(auctioneer-operation-at end-of-round generate-quote)

; This is a double auction so we allow multiple buyers
; and multiple sellers
(buyers many)
(sellers many)

; Pricing parameter- set clearing price halfway between
; the bid quote and the ask quote
(k-value 0.5)

; Allow agents to see each others' bids
(shouts-visible)

)

(defrule determine-clearing-price

; In a uniform-price k-CDA we determine clearing price
; based on the current market-quote and the value of k

(quote (ask ?ap) (bid ?bp))
(k-value ?k)

=>

(bind ?tp (transaction-price ?k ?ap ?bp))
(assert (transaction-price ?tp))
)
```

Fig. 9. Example specification of a uniform k-CDA.

software provides skeleton functionalities for implementing auctioneer agents, which we can extend to implement variants on common mechanisms. Building on JASA, we have implemented an auctioneer agent that is able to run auctions described using the ontological framework outlined above.

On the client side, we need to allow trading agents to place offers in the negotiation without hard-coding any knowledge of the rules of the mechanism. Trading agents aim at obtaining a good deal, but not necessarily the *best* possible deal, as mentioned above, and this requires strategic bidding. We already mentioned that strategic bidding through game-theoretic techniques is not practically computable for most non-trivial cases.

Therefore, we opt for equipping our agents with simple reinforcement learning capabilities that permit the agents to acquire reasonably good strategies after several iterations of play against a new mechanism and the agents trading therein.

The advantage of specifying the negotiation rules in an ontology, is that our agents can potentially make use of this information to select and configure an appropriate learning algorithm based on their knowledge of the mechanism. Rather than trying to reason about the *optimal* strategy to play for a particular mechanism, we use the ontological description of the mechanism to reason about which *adaptive* strategy to play.

We can divide adaptive negotiation strategies based on learning algorithms into three broad classes:

(1) Mimicry learning strategies, such as Cliff's ZIP strategy (Cliff and Bruten, 1997);
(2) Strategies based on myopic stimuli-response learning algorithms, such as the Roth–Erev algorithm (Erev and Roth, 1998);
(3) Strategies based on Markoff descision procedure (MDP) algorithms, such as the Q-learning algorithm.

Currently we are investigating whether it is possible to select the appropriate class of strategy based on knowledge of the negotiation mechanism. For example, strategies based on MDP learning are able to adapt different behaviour for different states of the negotiation. Therefore, a possible heuristic for selecting this class of algorithm is to determine whether or not the mechanism issues quote data, which can then be used to track the current state of the negotiation. The learning algorithm may then be able to adapt different policies for different situations, such as where we have a highly competitive market with a narrow bid-ask spread as opposed to an open market with a wide spread. Using our framework we are able to specify heuristics such as those in Fig. 10, which specifies that we should use an MDP strategy in double-sided mechanisms that issue quotes at every round.

```
(defrule tuning-quotevisible

(tuning-strategies)
(auctioneer-operation-at end-of-round generate-quote)
(sellers many)

=>

(assert
 (use-strategy "uk.ac.liv.auction.agent.MDPStrategy"))
)
```

Fig. 10. Heurstic for selecting a strategy based on an MDP learning algorithm.

7. Background

Research in automated negotiation (Rosenschein and Zlotkin, 1994; Kraus, 2001; Sandholm, 1999) to date has focused on the design of *negotiation protocols* specifically tailored to account for particular interactions among agents (Rosenschein and Zlotkin, 1994; Kraus, 1997, 2001; Jennings et al., 2001). and on the *strategies* that agents should employ in order to interact successfully according to one of these protocols.

Some prototypical standards for negotiation have been proposed. For example, the FIPA agent communication language (ACL) provides a number of performatives (message types) explicitly intended to support negotiation.[2] An example is the `cfp` (call for proposals) performative, intended to support contract-net style task sharing via negotiation (Smith, 1980). However, the FIPA performatives are intended to be used by agents *while negotiating*: they are not appropriate for defining the properties of negotiation protocols. In fact, there is currently no widely accepted standard for expressing different negotiation protocols.

Automated negotiation is particularly relevant in open environments such as the Internet, or, as it seems it will be, the Semantic Web (Berners-Lee, 1999; Fensel et al., 2001; Hendler, 2001). Explicit and machine understandable representation of the agents knowledge are the primary characteristic of the Semantic Web. According to Berners-Lee and colleagues, "*The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation*". Semantic Web technologies are broadening the horizon for many applications such as e-commerce, which require increasingly open and flexible architectures. In these environments, better cooperation between computers and people is achieved by reducing the limitations that are imposed on the agents and on the types of interaction they can be involved in. Thus, agents should be free to join and leave interactions at any time. However, this

[2]The Foundation for Intelligent Physical Agents. See `http://www.fipa.org/`.

means that a number of new challenges arise as illustrated by Willmott et al. (2002), where the authors mention, among other things, the agents ability to interact with multiple markets. This translates into a clear definition of the semantics associated with messages exchanged between agents and marketplaces and a clear definition of the rules which permit trading in the context of specific institutions.

From another perspective, the common representation of agents' capability to negotiate in open environments such as the Internet (or the Semantic Web), can be seen as an explicit representation of a service offered by an agent. Research on Web services and in particular on the explicit representation of service capabilities and processes has flourished in the past 4 years and it bears some similarities with the work presented in this paper. Research efforts such as OWL-S (www.daml.org/services/) build on top of OWL an upper ontology for services that essentially provides three types of knowledge about a service, namely: knowledge about what the service requires of agents and/or user, and about what the service provides, knowledge about how the service works and, finally, knowledge on how the service is used. In more detail, the description of what the service requires and provides is comparable to the "static notions" modelled in the shared ontology for negotiation, whereas the description of how a service works is similar to the description of the negotiation process itself, and indeed both base their process description on PSL.

The problem of modelling processes has also been addressed by research in problem-solving methods (PSMs), which aim to model knowledge concerning how to perform a task. This is made possible by introducing the notion of *method ontology*. A method ontology is an ontology of problem-solving knowledge units plus control knowledge.

The basic unit of problem solving knowledge is a mapping of the form (Chandrasekaran et al., 1998):

⟨conditions on problem state (including goals)⟩
⟨conditions on domain knowledge⟩
⟨conditions on data describing the problem instance⟩
→
⟨changes to problem solving states (including goal components)⟩

The mapping described above is not to be interpreted as a rule, but just as a Knowledge Level (Newell, 1982) description of a basic unit of problem-solving knowledge. However, these mappings are implemented as rules at implementation level.

By means of these mappings, control knowledge can be specified, either explicitly by using standard vocabulary of control, such as *sequential control*, *conditional branching*, *recursion*, *iteration*, etc., or implicitly, by the interaction of domain knowledge and the problem state.

In comparison with the use of PSL, the representation of processes as problem-solving methods requires a simpler representation of time and of the subactivity theory, since most of the protocol would be represented by means of cause-effect rules. On the other hand, few languages have been proposed to represent libraries of problem-solving methods, such as OCML (Motta, 1999) and UPML (Fensel et al., 1999). For UPML, some modelling support is provided in the form of a plugin for Protegé 2000 that express the modelling primitives used to represent a problem-solving method.

8. Conclusions

In this paper, we have presented an ontological approach to automated negotiation, particularly suited to open environments such as the Internet, or the Semantic Web. In this kind of environment fewer limitations should be imposed on the agents and on the types of interaction they can be involved in. Ideally, agents should be able to join and leave interactions at any time, and any agent intending to join an interaction should conform to the specific negotiation protocol which regulates that type of interaction.

In our approach, protocols are not coded within agents, as this would be a limitation. A new agent joining a pre-existing interaction should acquire the negotiation protocol governing that interaction from the marketplace itself, through an advertisement of the type of protocol used. In order to permit interoperability, the protocol is defined in terms of a *shared ontology* of negotiation which provides the basic vocabulary that agents must share in order to discuss the terms of the participation in the negotiation session.

The negotiation ontology that we have illustrated in Section 3 is intended to capture similarities between the different negotiation mechanisms. This kind of generic description can be used as classification framework that permits the analysis of the negotiation protocols available, and the development of new ones. But also, by committing to the same high level concepts, the communication of negotiation rules among the agents is facilitated, thus improving flexibility.

With respect to the use of the ontology as input for tuning the agent strategy to the particular type of protocol, this work attempts to characterise the complexity of the protocol defined by a given mechanism and to recommend a potential solution technique. For some mechanisms, it will be possible to find an optimal strategy using computational techniques. For more complex mechanisms, we have argued that game-theoretic solutions are computationally expensive; in cases as the ones described in Section 6 our approach can be used to recommend a strategy based on a class of learning algorithm.

Acknowledgements

This research was funded by Hewlett-Packard Labs, UK.

References

Bartolini, C., Preist, C., Jennings, N.R., 2002. A generic software framework for automated network. In: Proceedings of the First International Conference on Autonomous Agent and Multi-Agent Systems.

Berners-Lee, T., 1999. Weaving the Web. Orion Business, London.

Boley, H., Tabet, S., Wagner, G., 2001. Design rationale of RuleML: A markup language for semantic web rules. In: Proceedings of the International Semantic Web Working Symposium (SWWS), 2001.

Chalupsky, H., 2000. Ontomorph: a translation system for symbolic knowledge. In: Cohn, A.G., Giunchiglia, F., Selman, B. (Eds.), Principles of Knowledge Representation and Reasoning. Proceedings of the Seventh International Conference (KR'2000), San Francisco, CA. Morgan Kaufmann, Los Altos, CA, pp. 471–482.

Chandrasekaran, B., Josephson, J.R., Benjamins, V.R., 1998. Ontology of task and methods. In: Proceedings of the Eleventh Workshop on Knowledge Acquisition, Modelling and Management (KAW'98), Banff, Ala, Canada,. University of Calgary.

Cliff, D., Bruten, J., 1997. Minimal-intelligence agents for bargaining behaviours in market-based environments. Technical Report HPL-97-91, HP Labs.

DAML, 2001. The DARPA agent markup language. See http://www.daml.org/.

Decker, S., Melnik, S., van Harmelen, F., Fensel, D., Klein, M., Broekstra, J., Erdmann, M., Horrocks, I., 2000. The semantic web: the roles of XML and RDF. IEEE Internet Computing 4 (5), 63–74.

Erev, I., Roth, A.E., 1998. Predicting how people play games: reinforcement learning in experimental games with unique, mixed strategy equilibria. American Economic Review 88 (4), 848–881.

Fensel, D., Benjamins, V.R., Motta, E., Wielinga, B.J., 1999. UPML: a framework for knowledge system reuse. In: IJCAI. pp. 16–23.

Fensel, D., Horrocks, I., van Harmelen, F., McGuinness, D.L., Patel-Scheider, P.F., 2001. The semantic web. IEEE Intelligent Systems 16 (2), 24–25.

Field, S., Stolze, M., Stroebel, M., 2000. The London Classification: 1st e-negotiations Workshop—Negotiations Beyond Price. In: DEXA Workshops.

Fridman Noy, N., Fergerson, R.W., Musen, M.A., 2000. The knowledge model of protege—2000: combining interoperability and flexibility. In: Dieng, R. (Ed.), Proceedings of the 12th EKAW Conference, vol. LNAI 1937. Springer, Berlin, pp. 17–32.

Genesereth, M.R., Fikes, R., Bobrow, D., Brachman, R., Gruber, T., Hayes, P., Letsinger, R., Lifschitz, V., MacGregor, R., McCarthy, J., Norvig, P., Patil, R., Schubert, L., 1992. Knowledge interchange format. version 3.0. reference manual. Technical Report, Stanford University. http://www-ksl.stanford.edu/knowledge-sharing/kif/.

Grosof, B.N., Gandhe, M.D., Finin, T.W., Sweetjess: Inferencing in situated courteous ruleml via translation to and from jess rules. In: Proceedings of the ISWC '02 International Workshop on Rule Markup Languages for Business Rules on the Semantic Web, June 2002, Sardinia (Italy).

Gruber, T.R., 1993. A translation approach to portable ontology specifications. Knowledge Acquisition 5 (2), 199–220.

Gruninger, M., 2003. Ontology of the process specification language. In: Handbook of ontologies.

Haugen, R., McCarthy, W.E., 2000. Rea: A semantic model for internet supply chain collaboration. In: The ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, Business Object Component Design and Implementation Workshop VI: Enterprise Application Integration.

Hendler, J., 2001. Agents and the semantic web. IEEE Intelligent Systems 16 (2), 30–37.

Horrocks, I., Patel-Schneider, P.F., van Harmelen, F., 2002. Reviewing the design of DAML+OIL: an ontology language for the semantic web. In: Proceedings of the 18th National Conference on Artificial Intelligence (AAAI 2002).

Horrocks, I., et al., Swrl: a semantic web rule language combining owl and ruleml. http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/.

Jennings, N.R., Faratin, P., Lomuscio, A.R., Parsons, S., Sierra, C., Wooldridge, M., 2001. Automated negotiation: prospects methods and challenges. International Journal of Group Decision and Negotiation 10 (2), 199–215.

Kraus, S., 1997. Negotiation and cooperation in multi-agent environments. Artificial Intelligence 94 (1–2), 79–98.

Kraus, S., 2001. Strategic Negotiation in Multiagent Environments. The MIT Press, Cambridge, MA.

Laera, L., Tamma, V., Bench-Capon, T., Semeraro, G., 2004. Sweetprolog: a system to integrate ontologies and rules. In: Proceedings of the ISWC'04 workshop on Rules and Rule Markup Languages for the Semantic Web.

Lassila, O., McGuinness, D., 2001. The role of frame-based representation on the semantic web. Knowledge Systems Laboratory Report KSL-01-02, Stanford University, 2001.

Lomuscio, A.R., Wooldridge, M., Jennings, N.R., 2000. A classification scheme for negotiation in electronic commerce. In: Dignum, F., Sierra, C. (Eds.), Agent Mediated Electronic Commerce: a European Perspecive. Springer, Berlin, pp. 19–33.

Motta, E., 1999. Reusable Components for Knowledge Modelling: Principles and Case Studies in Parametric Design. IOS Press.

Neches, R., Fikes, R.E., Finin, T., Gruber, T.R., Patil, R., Senator, T., Swartout, W.R., 1991. Enabling technology for knowledge sharing. AI Magazine 12 (3), 36–56.

Newell, A., 1982. The knowledge level. Artificial Intelligence 18 (1), 87–127.

OWL, web ontology language. http://www.w3.org.

Phelps, S., Tamma, V., Wooldridge, M., Dickinson, I., 2004. Toward open negotiation. IEEE Internet Computing 8 (2), 70–75.

Rosenschein, J.S., Zlotkin, G., 1994. Rules of Encounter: Designing Conventions for Automated Negotiation among Computers. The MIT Press, Cambridge, MA.

Sandholm, T., 1999. Distributed rational decision making. In: Weiß, G. (Ed.), Multiagent Systems. The MIT Press, Cambridge, MA, pp. 201–258.

Smith, R.G., 1980. A Framework for Distributed Problem Solving. UMI Research Press.

Studer, R., Benjamins, V.R., Fensel, D., 1998. Knowledge engineering, principles and methods. Data and Knowledge Engineering 25 (1–2), 161–197.

Tesfatsion, L., 2001. Agent-based computational economics: growing economies from the bottom up. Technical Report, Iowa State University.

Willmott, S., Calisti, M., Rollon, E., 2002. Challenges in large-scale open agent mediated economies. In: Padget, J., Parkes, D.C., Sadeh, N.M., Shehory, O., Walsh, W.E. (Eds.), Proceedings of the AAMAS'02 Workshop on Agent Mediated Electronic Commerce IV: Designing Mechanisms and Systems.

Wurman, P.R., Wellman, M.P., Welsh, W.E., 2001. A parametrization of the auction design space. Games and Economic Behavior 35 (1/2), 304–338.

XML, 2001. The xtensible markup language. See http://www.xml.org/.