DR-BROKERING – A Defeasible Logic-Based System for Semantic Brokering

Grigoris Antoniou[1] Thomas Skylogiannis [2]
1 Institute of Computer Science, FORTH, Greece
antoniou@ics.forth.gr
2 Department of Computer Science, University of Crete, Greece
dogjohn@csd.uoc.gr

Antonis Bikakis[3] Nick Bassiliades[4]
3 Department of Computer Science, University of Crete, Greece
bikakis@csd.uoc.gr
4 Department of Informatics, Aristotle University of Thessaloniki, Greece
nbassili@csd.auth.gr

Abstract

Electronic Brokering, is a good candidate for taking up Semantic Web technology. In this paper we study the brokering and matchmaking problem that is, how a requester's requirements and preferences can be matched against a set of offerings collected by a broker. The proposed solution uses the Semantic Web standard of RDF to represent the offerings, and a deductive logical language, based on non-monotonic reasoning, for expressing the requirements and preferences. We motivate and explain the approach we propose, and report on a prototypical implementation exhibiting the described functionality, in JADE agent environment.

1. Introduction

E-Commerce describes the revolution that is currently transforming the way business is conducted, through the use of information technology, and in particular the World Wide Web. According to [14], in the 1st generation e-Commerce applications (current state), buyers and sellers are humans who typically browse through a catalogue of well-defined commodities (e.g. flights, books…) and make fixed price purchases usually by means of credit card transaction. Humans are in the loop of all stages of buying process, something which is time consuming.

The 2nd generation of e-Commerce will be realized through the use of automated methods of information technology. Web users will be represented by software agents. According to [17], there is an increasing use of software agents for all the aspects of e-Commerce.

This vision is consistent with the Semantic Web initiative [6], which enriches the current Web through the use of machine-processable information about the meaning (semantics) of information content. This way, the meaning of displayed information is accessible not only to humans, but becomes also accessible to software agents.

At the present work we deal with semantic-based brokering systems which help both service providers and requesters to match their interests. The key operations in such systems are to:

1. Identify appropriate services that satisfy user requirements;
2. Select the best service(s) based on the user preferences.

How to address these questions using Semantic Web technology, is the main focus of the present work. The three basic roles that we identify are the service requester (or the buyer), the service provider (or seller), and the broker. The technical solution we provide is based on the following key ideas:

- Service requesters, service providers and brokers are represented by software agents.
- The requirements of the service requester are represented in a logical language using rules and priorities. These requirements include both indispensable requirements that must be met for a service to be acceptable (for example, air-conditioning is required), and soft requirements (preferences) that can be used to select among the potentially acceptable offerings. These requirements are communicated to the broker

agent by the requester agent. This communication initiates a brokering activity.

- The offerings are represented in a certain semi-structured format using the Semantic Web standard language RDF [4] for describing Web resources. The provider agents communicate the offerings to the broker agent.
- The terminology shared by providers, requesters and brokers is organized in ontologies using the Semantic Web standard of RDF Schema [7].
- The broker is also a software agent and has special knowledge both for the declarative language and the advertisement format. It also has the ability to perform semantic checks to the information it receives.
- When the broker receives a request it matches the request to the offerings by running the request specification against the available offerings, making use of information provided by the shared ontology, as required. Then the requester's preferences are applied to select the most suitable offering(s) which are then presented to the requester.

The remainder of the paper is organized as follows. Section 2 describes our solution to the brokering problem, including a rationale for the chosen technologies. Section 3 illustrates the approach using a concrete example. Section 4 describes the technical details of a system that implements the solution. Finally, section 5 reviews related work and section 6 concludes the paper and poses future research directions.

2. Semantic Web-Enabled Brokering and Matchmaking

2.1 Brokering and Matchmaking Architectures

Middle agents are special purpose agents which help other agents to find each other or delegate their requested services As described in [25], three different kinds of middle agents prevail. They are called *matchmakers* (or Yellow Pages Services), *facilitators* and *brokers* respectively. We borrow the next two figures from their work. A typical architecture of a matchmaker is depicted in Fig.1. Different service providers advertise their capabilities (1) and the matchmaker puts them into a repository. When the matchmaker is asked for a particular service by a service requester (2), it returns information about all the available service providers (3).

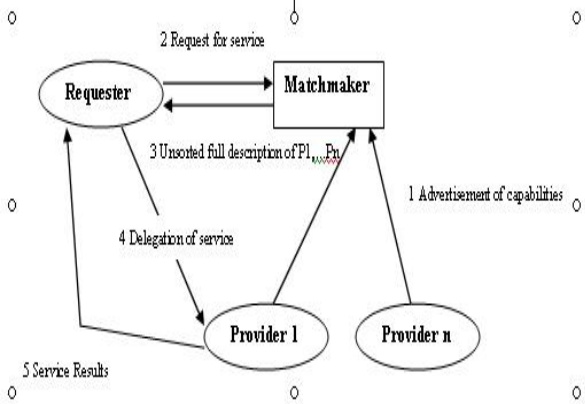

Fig. 1Matchmaker Architecture

It now depends on the requester, which provider it will choose (4) for the required service. Lastly, the provider serves the request and returns the results (5). It is assumed that the "address" of a matchmaker is well-known.

Facilitators operate in a slightly different way as we can see in Fig.2 Initially, providers advertise their capabilities (1). After requesters have located a facilitator (perhaps by means of matchmaker), they pass on their preferences along with the delegation of a service (2). The facilitator, in turn, picks one of the providers to delegate the requested service (3). The provider then returns the result (4) and the facilitator returns it to the requester (5).

A variation of this architecture could be that the facilitator agent itself performs the serving of a request using services and information from other agents in conjunction with his own services. In the latter case the middle agent is called "broker". We use the latter variation for our implementation. However, we would like to stress that our technology can easily be adapted to realize any of the above architectures; we have chosen to implement the broker architecture to demonstrate the feasibility of the overall approach.

2.2 Description of Offerings

The offerings are described in RDF, the standard Semantic Web language for representing factual statements. This choice supports interoperability among agents and applications.

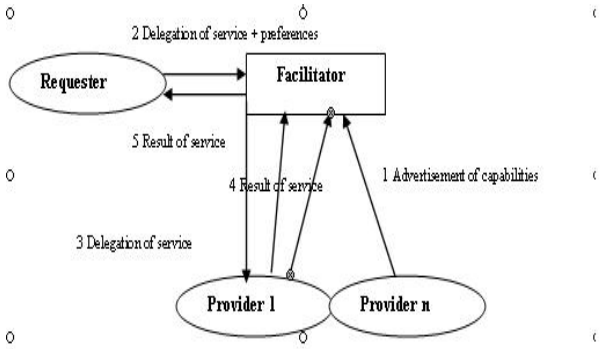

Fig. 2 Facilitator-Broker Architecture

The offerings are enriched through reference to a shared ontology. We assume that this ontology is expressed in RDF Schema, a simple ontology language based on RDF. We have chosen this language over the use of OWL [13] because at present it is not clear how the deductive capabilities of OWL and rule systems can be combined; it is one of the main research lines in the Semantic Web community. We could certainly use most features of OWL Lite, given that they can be expressed using rules [12].

2.3 Description of Requests and Preferences

The requirements and preferences of the requester are described in a logical language. Before choosing one or several languages for the specification of requests it is important to establish a set of criteria that such languages need to satisfy. The criteria presented below are inspired from those formulated by [15] in the context of techniques for information modeling. They encompass several well-known principles of language design.

Firstly, a language for specifying requirements and preferences needs to be *formal*, in the sense that its syntax and its semantics should be precisely defined. Secondly, the language should be *conceptual*. This, following the well-known *Conceptualization Principle* of [11], effectively means that it should allow its users to focus only and exclusively on aspects related to requirements, without having to deal with any aspects related to their realization or implementation. Thirdly, in order to ease the interpretation of strategies and to facilitate their documentation, the language should be *comprehensible*. Comprehensibility can be achieved by offering a graphical representation, by ensuring that the formal and intuitive meanings are as much in line as possible, and by offering structuring mechanisms (e.g. decomposition). These structuring mechanisms often lead to *modularity*, which in our setting means that a slight modification to a strategy should concern only a specific part of its specification. As we are interested in the automation of the brokering process, the requirements description language should be *executable.* Finally, the language that we aim should be sufficiently *expressive*, that is, it should be able to precisely capture a wide spectrum of requirements.

We have chosen *defeasible logic* to represent requesters' requirements and preferences because it satisfies the above criteria. In particular,

- It is a formal language with well-understood meaning ([2] presents a proof theory, [19] its model semantics, and [10] its argumentation semantics), thus it is also predictable and explainable.
- It is designed to be executable; implementations are described in [20]. It is also scalable, as demonstrated in the same paper, where it was shown that 100,000 rules can be processed efficiently. This is so because the computational complexity of defeasible logic is low [18].
- It is expressive, as demonstrated by the use of rules in various areas of information technology. In fact, among the logical systems, it is rule-based systems that have been best integrated in mainstream IT.
- Finally, it is suitable for expressing requirements and preferences in our setting. This is so because it supports the natural representation of important features:
  - Rules with exceptions are a useful feature in our problem. For example, a general rule may specify acceptable offerings, while more specific rules may describe cases in which the general rule should not apply and the offering should not be accepted. We will elaborate on this point in the next section when we consider a concrete example.
  - Priorities are an integral part of defeasible logic, and are useful for expressing user preferences for selecting the most appropriate offerings from the set of the acceptable offerings.

3. A Concrete Example

3.1 The Scenario

Bob, who holds a middle management position, looks for an appropriate hotel room for his business trip to Athens. He wishes to stay at a central hotel, or at least at a hotel close to public transport. And he requires the hotel to have air-conditioning, a gym, and generally speaking a business standard.

In accordance with tight budgeting rules of his company, Bob is willing to pay a modest price: 70 Euros per night. However, if the hotel is central he is willing to pay 80 Euros, and if the hotel has a pool he is willing to pay 90 Euros.

If given the choice, he would go for a hotel with a central location, with the lower price being his secondary preference criterion.

3.2 Formalization of Requirements

We show how Bob's firm requirements are represented in defeasible logic. The predicate *acceptable(X)* is used to denote that a hotel is acceptable. The first rule says that, a priori, all hotels are acceptable.

r_1: $\Rightarrow$ acceptable(X)

However, any hotel not satisfying one of the required features is unacceptable. The following rules describe exceptions to the first, more general rule. In fact, rules with exceptions are a common, very useful representational mechanism of defeasible logic. Note that the exception rules are declared to be stronger than the general rule.

r_2: ¬central(X), ¬publicTransport(X) $\Rightarrow$ ¬acceptable(X)
r_3: ¬gym(X)$\Rightarrow$ ¬acceptable(X)
r_4: ¬aircon(X) $\Rightarrow$ ¬acceptable(X)
r_5: ¬businessStandard(X) $\Rightarrow$ ¬acceptable(X)
$r_2 > r_1$, $r_3 > r_1$, $r_4 > r_1$, $r_5 > r_1$

Next we must represent the price Bob is willing to pay at most. The predicate *offer(X,Y)* denotes that Bob is willing to pay at most Y Euros for hotel X.

r_6: $\Rightarrow$ offer(X,70)
r_7: central(X) $\Rightarrow$ offer(X,80)
r_8: pool(X) $\Rightarrow$ offer(X,90)
$r_8 > r_7 > r_6$

A hotel is unacceptable if its price is higher than what Bob is willing to pay. This rule is also an exception to the general rule r_1.

r_9: price(X,Y), offer(X,Z), X>Z $\Rightarrow$ ¬acceptable(X)
$r_9 > r_1$

3.3 Representation of Offered Hotels

The hotel offerings maintained by the broker are stored as RDF facts, and are processed by the rules representing the requirements. To increase readability we also show the offerings as logical facts. For example, hotel h1 can be described by the facts:¬central(h1), aircon(h1),publicTransport(h1), category(h1,2),gym(h1), price(h1,50), ¬pool(h1). Table 1 shows seven offerings in table form. It is interesting to look at the category information which does not appear in the rules describing Bob's requirements. This is natural since Bob, coming from a different country, does not know the meaning of the hotel ratings in Greece. It is an ontology of the tourism domain that would establish a link between the two. In our example, we assume that business standard is provided by Greek hotels with at least three stars.

Based on the rules of section 3.2 and the hotel offerings, we see that: a) Hotel h1 is unacceptable because it does not provide business standard (rule r_2).b)Hotel h4 is unacceptable because it does not have air-conditioning (rule r_4). c) Hotel h6 is unacceptable because it does not have a gym (rule r_3). d) Hotel h2 is unacceptable because its price (100) is higher than what Bob is willing to pay (90; rules r_8, r_9).e) Hotels h3, h5 and h7 are acceptable (rule r_1).

Hotel	**h1**	**h2**	**h3**	**h4**	**h5**	**h6**	**h7**
Central	No	Yes	Yes	Yes	No	No	Yes
Public Transport	Yes	No	Yes	No	Yes	Yes	Yes
Gym	Yes	Yes	Yes	Yes	Yes	No	Yes
Pool	No	Yes	Yes	Yes	No	No	No
A/C	Yes	Yes	Yes	No	Yes	Yes	Yes
Category	2	4	3	3	3	3	3
Price	50	100	80	70	60	50	60

Table 1: A Set of offered hotels

4. System Implementation

4.1 Agent Framework

The agent framework we used for the development of our system is JADE [5]. JADE is an open-source middleware for the development of distributed multi-agent applications, based on the peer-to-peer communication architecture. JADE is Java-based and compliant with the FIPA specification [9]. It provides libraries for agent discovery, communication and interaction, based on FIPA standards.

4.2 System Architecture and Modules

The system architecture is shown in Fig. 3. In the following we describe the modules of this architecture. The main modules of the architecture are the reasoning module (R.M.), the control module (C.M.) and the internet module (I.M.). Each of the sub-modules that are described below, belongs to a particular main module, as depicted in Fig. 3.

RDF translator

The role of the RDF translator is to transform the RDF statements into logical facts, and the RDFS statements into logical facts and rules. This transformation allows the RDF/S information to be processed by the rules provided by the Service Requester (representing the requester's requirements and preferences).

For RDF data, the SWI-Prolog RDF parser is used to transform them into an intermediate format, representing triples as *rdf(Subject, Predicate, Object)*. Some additional processing (i) transforms the facts further into the format *Predicate(Subject, Object)*; (ii) cuts the namespaces and the "comment" elements of the RDF files, except for resources which refer to the RDF Schema, for which namespace information is retained.

In addition, for processing RDF Schema information, the following rules capturing the semantics of RDF Schema constructs are created:

A: C(X):- rdf:type(X,C).
B:C(X):- rdfs:subClassOf(Sc,C),Sc(X).
C:P(X,Y):-rdfs:subPropertyOf(Sp,P),Sp(X,Z).
D: D(X):- rdfs:domain(P,D),P(X,Z).
E: R(Z):- rdfs:range(P,R),P(X,Z).

Let us consider rule B that captures the meaning of the subclass relation of RDFS. A class Sc is subclass of a class C when all instances of Sc are also instances of C. Stated another way, if X is an instance of Sc then it is also instance of C. That is exactly what rule B says.

All the above rules are created at compile-time, i.e. before the actual querying takes place. Therefore, the above rules although at first they seem second-order, because they contain variables in place of predicate names, they are actually first-order rules, i.e. predicate names are constant at run-time.

Rule Parser & Translator

The Rule Parser is responsible for checking the validity of the defeasible rules, which are submitted by the Service Requester. The rules are considered to be valid, if they follow the standard syntax of defeasible logic, as described in [2]. If there are syntax errors, the system informs the user about these errors, and does not proceed to the translation. Otherwise, the parser creates a symbol table, which includes all the rules and priority information, and passes this table to the Translator.

The Rule Translator is responsible for transforming the rules submitted by the Service Requester using the syntax of defeasible logic, into Prolog rules that emulate the semantics of defeasible logic. The method we use for translating defeasible theories into logical programs is described in detail in [3].

The logical program that derives from this procedure will be later combined with the logical facts that represent the RDF triples, and will be used to evaluate the queries of the Service Requester.

Query Translator

In order to apply a query to the Prolog files, which contain the rules and the facts, it must be properly transformed into a valid Prolog query. This task is performed by the Query Translator. There is a standard format for the queries that the Service Requester can make:

D x : which are the literals (atoms or their negation) x which are provable according to the rules provided by the Service Requester.

The literals 'x' represent the conclusions of the rules, which are submitted by the Service requester. 'x' can be for example of the form '*accept_hotel(X)*'. In this case a query of the form '*D accept_hotel(X)*', is intended to find those literals X satisfying the conclusion '*accept_hotel(X)*'.

Rule-Query-RDF Loader

The role of this module is to download the files which contain the rules and the query of the user, in defeasible logic format. It also loads the appropriate RDF data which correspond to service provider's advertisements.

Semantic-Syntactic Validator

This module is an embedded version of [24], a parser for validating RDF documents. Upon receipt of an advertisement, the RDF document which corresponds to that advertisement is checked by this module. Among others, the tests that are being performed are: class hierarchy loops, property hierarchy loops, domain/range of subproperties, source/target resources of properties and types of resources.

Interaction and Communication Modules

For the implementation of these modules, we used the Java classes of JADE framework. The communication module is responsible for sensing the network and notifying the control module when an external event (e.g. a request message) occurs. In order to decide the course of action based on with the incoming messages, the agent uses its interaction module, which defines the allowed sequence of actions, according to the used interaction protocol, which in our case is the FIPA request interaction protocol.

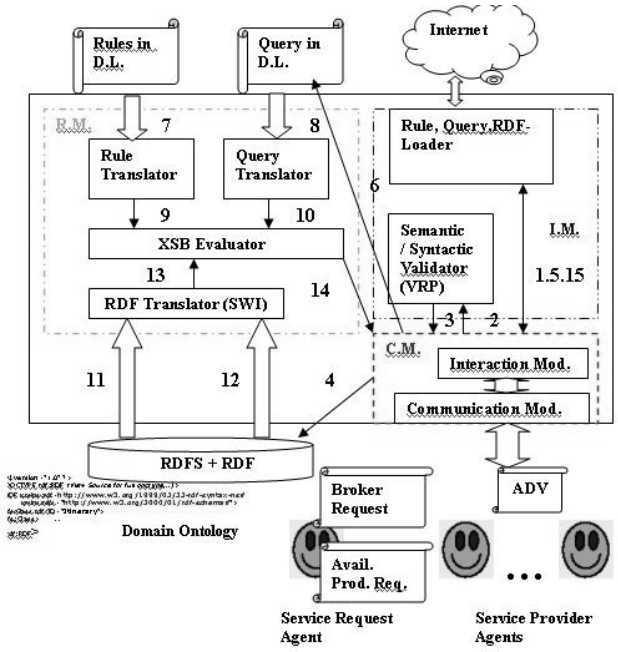

Fig.3 Agent-Based Semantic Brokering Architecture

XSB Evaluator

The role of the Evaluator is to apply the queries to the Prolog files, which contain the facts and the rules, and to evaluate the answer. When the Service Requester makes a query, the Evaluator compiles the files containing the facts and the rules, and applies the transformed Prolog query to the compiled files.

The answer of the query is sent to the Control Module of the system. We have employed XSB Prolog as the compiler and evaluator for our system. We made this choice, as we needed a logic programming system supporting well-founded semantics. XSB Prolog offers this functionality through its *tnot* operator.

4.3 A Typical Trace

Fig 4. shows a typical trace which we briefly discuss. In this trace we assume an environment in which broker agents must register with a Yellow Page, or directory facilitator, service so that they can be located by potential requesters. Similarly, requesters locate a suitable broker via visiting the directory facilitator (REQUEST:0,INFORM:0,REQUEST:1,INFORM:1,REQUEST:1,INFORM:1).

Seller in turn directly accesses the broker and sends an advertisement (REQUEST:2). Broker informs the seller agent that it will service its request (AGREE:2). It then extracts the field "ITEM_rdfInfoAtWeb" from the message and passes the URL to the Rule-Query-RDF Loader, which downloads the RDF document from the Internet (step 1 in Figure 6). Afterwards, the semantic/syntactic validator module is fed with the RDF document (step 2 in Figure 6). Finally, the validator returns the result to the control module (step 3 in Figure 6) of the broker, which either returns to the seller a result of success (INFORM:2) or a message of type ADVERTISE_ERROR. The RDF file is stored in a local database (step 4 in Figure 6).

At a later time, a service requester searches the directory facilitator for a brokering service (REQUEST:3) and receives an answer (INFORM:3) suggesting a broker. The service requester agent asks the broker to inform it with the available categories of products for brokering (REQUEST:4). The broker informs that it will service its request (AGREE:4) and, when ready, returns the result (INFORM:4). After the requester is notified of the available categories, it issues a broker request (REQUEST:5) in the category of interest (e.g. "itinerary"). This message provides information about the location of the rules, the rules language, and the query that the buyer wants to ask the broker. The rules and the query are downloaded (step 5

in Fig. 3) and stored locally (step 6 in Fig. 3). Then, the rules and the query, which are in defeasible logic, are fed into the rule and query translator modules (steps 7, 8 in Fig. 3) and then into XSB engine (steps 9, 10 in Fig. 3).

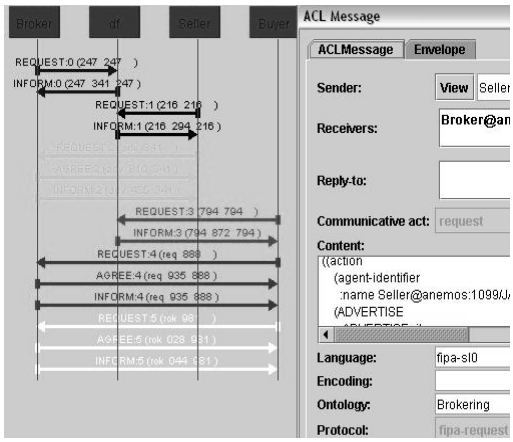

Fig. 4 Trace From the Semantic Brokering Interactions

Also, the instances and the domain ontology are transformed to facts (steps 11, 12 in Fig. 3) and fed into the XSB engine (step 13 in Fig. 3). All the transformed data are combined by XSB and the result is returned to the control module (step 14 in Fig. 3). Finally, the broker sends to the buyer an inform message (INFORM:5) which contains a link to the result of the query, which is the address of one of the service providers and the coresponding RDF description.

5. Related Work

InfoSleuth [21] is an agent-based information discovery and retrieval system that adopts "broker agents" to perform the syntactic and semantic matchmaking. The broker uses a rule-based reasoning engine, implemented in LDL The brokering is realized in two levels. Syntactic brokering and. Semantic brokering

[23] proposes the use of RDF/RDFS for the matchmaking process. Each advertisement, either for service request, or for service offering is represented as an RDF resource. Properties from this resource characterize the type of requested or offered service. The advertisements are stored into a repository and the matching of advertisements is reduced to matching of RDF graphs. The authors implement a matching algorithm.

[16] assesses the requirements for a service description language and ontology, and argue that DAML+OIL and DAML-S common service ontology, fulfil these requirements. This argument is supported by their design and implementation of a prototype matchmaker which uses a description logic reasoner to match service advertisements and requests based on the semantics of ontology based service descriptions. Similar is the work of [22]. They also use DAML-S to describe the advertisements along with the request and afterwards they use a matching algorithm.

[8] propose the iAgent which consists of inference, control and communication layer. They propose that the facts are extracted from semantic mark-up documents that are written in DAML+OIL. The fact translator module of the iAgent, converts all the DAML+OIL documents into prolog format. iAgent uses a Horn-based logic engine (SWI-Prolog) for inferencing.

6. Conclusion and Future Work

In this paper we studied the brokering and matchmaking problem, that is, how a requester's requirements and preferences can be matched against a set of offerings collected by a broker. The proposed solution uses the Semantic Web standard of RDF to represent the offerings, and a deductive logical language for expressing the requirements and preferences. We motivated and explained the approach we propose, and reported on a prototypical implementation exhibiting the described functionality in an agent environment.

In the future we intend to extend the described work in various directions:

Firstly, we observe that the broker maintains the offered services locally, but in our system we did not present technological support for maintaining the offerings in a database system. We currently work on the use of use RDF Suite [1], a system for storing and retrieving RDF/S information. The extension of the system is straightforward.

Another idea worth following is to represent the offerings using rules, too. This way we can provide richer representation capabilities than are possible with RDF, e.g. to describe conditional capabilities (that is, certain properties hold under specific conditions).

We also plan to implement the matchmaker and facilitator functionalities of [25] outlined in section 2. Finally, we intend to integrate our brokering system, with the negotiation system described in [26]. This problem is orthogonal to that of brokering/matchmaking; for example, bargaining can

take place after identification of appropriate service providers.

References

[1] S. Alexaki, V. Christophides, G. Karvounarakis, D. Plexousakis and K. Tolle (2001). "The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases". In *Proc. 2nd International Workshop on the Semantic Web*, Hongkong, May 1, 2001

[2] G. Antoniou, D. Billington, G. Governatori and M.J. Maher (2001). "Representation results for defeasible logic". ACM Transactions on Computational Logic 2, 2 (2001): 255 - 287

[3] Antoniou & Bikakis (2004). "A System for Non-Monotonic Rules on the Web" .Workshop on Rules and Rule Markup Languages for the Semantic Web (RuleML 2004), G. Antoniou, H. Boley (Ed.), Springer-Verlag, Hiroshima, Japan, 8 Nov. 2004

[4] D. Beckett (2004). RDF/XML Syntax Specification, W3C Recommendation, February 2004. Available at: http://www.w3.org/TR/2004/REC-rdf-syntax-grammar-20040210/

[5] F.Bellifemine, G Caire, A.Poggi, G. Rimassa (2003). "JADE A White Paper". Telecom Italia EXP magazine Vol 3, No 3 September 2003

[6] T. Berners-Lee (1999). "*Weaving the Web"*. Harper 1999

[7] D. Brickley, R.V. Guha (2004). RDF Vocabulary Description Language 1.0: RDF Schema W3C Recommendation, February 2004. Available at: http://www.w3.org/TR/2004/REC-rdf-schema-20040210/

[8] Y-C. Chen, W-T. Hsu, P-H. Hung (2003). "Towards Web Automation by Integrating Semantic Web and Web Services". In *Proc. 12th International WWW Conference*

[9] FIPA. http://www.fipa.org

[10] G. Governatori and M.J. Maher. "An argumentation-theoretic characterization of defeasible logic". In Proceedings of the 14th European Conference on Artificial Intelligence}, Amsterdam, 2000. IOS Press

[11] J.J. van Griethuysen, editor. "*Concepts and Terminology for the Conceptual Schema and the Information Base"*. Publ. nr. ISO/TC97/SC5/WG3-N695, ANSI, 11 West 42nd Street, New York, NY 10036, 1982

[12] B. N. Grosof, I. Horrocks, R. Volz and S. Decker (2003). "Description Logic Programs: Combining Logic Programs with Description Logic". In: *Proc. 12th Intl. Conf. on the World Wide Web (WWW-2003),* ACM Press

[13] D.L. McGuinness , F. van Harmelen (2004). OWL Web Ontology Language Overview W3C Recommendation, February 2004. Available at: http://www.w3.org/TR/owl-features/

[14] M. He, N.R. Jennings, and H-F. Leung (2003). "On Agent-Mediated Electronic Commerce". *IEEE Transactions on Knowledge and Data Engineering* Vol. 15, No 4 July/August 2003

[15] A.H.M. ter Hofstede. "*Information Modelling in Data Intensive Domains"*. PhD thesis, University of Nijmegen, Nijmegen, The Netherlands, 1993

[16] L. Li and I. Horrocks (2003). "A Software Framework for Matchmaking Based on Semantic Web Technology". In *Proc. 12th International Conference on WWW*, ACM 2003

[17] P. Maes, R.H. Guttman and A.G. Moukas (1999). "Agents that Buy and Sell". *Communications of the ACM* Vol. 42, No. 3 March 1999

[18] M.J. Maher (2001). "Propositional Defeasible Logic has Linear Complexity". Theory and Practice of Logic Programming 1,6, p. 691-711

[19] M.J. Maher (2002). "A Model-Theoretic Semantics for Defeasible Logic", *Proc. Workshop on Paraconsistent Computational Logic*, 67 - 80, 2002

[20] M.J. Maher, A. Rock, G. Antoniou, D. Billington and T. Miller (2001). "Efficient Defeasible Reasoning Systems". *International Journal of Tools with Artificial Intelligence* 10,4 (2001): 483--501

[21] M. Nodine, W. Bohrer, A. Hee Hiong Ngu (1998). "Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth". In *Proc. 15th International Conference on Data Engineering*, IEEE Computer Society

[22] M. Paolucci, T. Kawamura, T.R. Payne, K. Sycara (2002). "Semantic Matching of Web Services Capabilities". In *Proc. 1st International Semantic Web Conference (ISWC-2002)*

[23] D. Trastour, C. Bartolini, J. Gonzalez- Castillo (2001). "A Semantic Web Approach to Service Description for Matchmaking of Services". HP Technical Report. August 2001

[24] VRP. The ICS-FORTH Validating Rdf Parser –VRP (2004). Available at: http://139.91.183.30:9090/RDF/VRP/

[25] H-C. Wong and K. Sycara (2000). "A Taxonomy of Middle-agents for the Internet". In *Proc. 4th International Conference on Multi Agent Systems (ICMAS-2000)*

[26] T. Skylogiannis, G. Antoniou, N. Bassiliades, G. Governatori (2004) "DR-NEGOTIATE - A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies". Submitted at the 5th International Conference on e-Technology, e-Commerce, e-Service (EEE05)