A Trusted Defeasible Reasoning Service for Brokering Agents in the Semantic Web

Kalliopi Kravari, Efstratios Kontopoulos and Nick Bassiliades

Dept. of Informatics, Aristotle University of Thessaloniki, GR-54124 Thessaloniki, Greece

{kkravari, skontopo, nbassili}@csd.auth.gr

Abstract. Based on the plethora of proposals and standards for logic- and rule-based reasoning for the Semantic Web (SW), a key factor for the success of SW agents is interoperability of reasoning tasks. This paper reports on the first steps towards a framework for interoperable reasoning among agents in the SW that deploys third-party trusted reasoning services. This way, agents can exchange their arguments, without the need to conform to a common rule or logic paradigm – via an external reasoning service, the receiving agent can grasp the semantics of the received rule set. The paper presents how a multi-agent system was extended with a third-party trusted defeasible reasoning service, which offers agents the ability of reasoning with incomplete and inconsistent information. In addition, a brokering trade scenario is presented that illustrates the usability of the approach.

1. Introduction

The *Semantic Web* (*SW*) [1] is a rapidly evolving extension of the WWW, in which the semantics of information and services is well-defined, making it possible for people and machines to use Web content and services for understanding and satisfying their requests. SW research is currently focusing on logic, reasoning and proof. *Intelligent agents* (*IAs*) can be highly favored by SW technologies [2], because of the interoperability the latter promises. The integration of *multi-agent systems* (*MAS*) with SW technology will significantly affect the use of the Web as we know it today; its next generation will feature groups of intercommunicating agents traversing it and performing complex actions on behalf of their users.

A core setback in agent interoperation is the variety in representation and reasoning. Despite KIF's efforts [3], there is still no globally agreed knowledge representation and reasoning formalism for agents. For SW agents, on the other hand, we can safely assume that *OWL* could be the global knowledge exchange language. As for rule- and logic-based reasoning, there is a variety of proposals and standards [4, 5, 6]. Thus, a key factor for the success of SW information systems

in general and agents in particular is reasoning task interoperability among multiple, heterogeneous web entities exchanging rule bases to justify their positions.

Reasoning interoperability among agents can be often (but not always) achieved by translating the received rule set into the receiving agent's formalism. This can only be accomplished, when the two agents use the same rule formalism with different syntax, or when the one formalism can be semantically translated into the other (e.g. translation between defeasible logic and Datalog rules [7]).

On the other hand, this paper proposes a simpler approach that does not rely on semantic interoperability, but on exchanging the rule base model, instead. This way, agents can exchange their position arguments, without the need to conform to a common rule paradigm or logic. The receiving agent can use an external reasoning service to grasp the semantics of the rule set, i.e. the result set of the received rule base. A critical assumption, of course, is that reasoning services are trusted and are hosted by authoritative organizations, like *W3C* or *RuleML.org*.

More specifically, this paper presents how a JADE MAS was extended with *defeasible reasoning* (*DR*) [8], i.e., the ability to reason with incomplete and inconsistent information. A DR service accompanied by a reputation mechanism was implemented as a JADE agent. The approach is generic, since any reasoner can be deployed as an agent-service in the system. Moreover, the paper presents a use case brokering trade scenario that illustrates the usability of these technologies.

In the rest of the paper, the implemented MAS is presented, focusing on the reasoning service that is based on DR-DEVICE, the core reasoning engine deployed in this paper. Section 3 presents the use case scenario and the paper is concluded with final remarks and directions for future work.

2. Defeasible Reasoning Service Agents

Defeasible reasoning constitutes a simple rule-based approach for efficient reasoning with incomplete and inconsistent information [8]. Its main advantages are enhanced representational capabilities and low computational complexity.

The DR-DEVICE [7] defeasible logic reasoner employs an OO RDF data model that treats properties as encapsulated attributes of resource objects, providing more compact representation and increased query performance. DR-DEVICE supports a rule language that extends RuleML with rule types, superiority relations among rules and conflicting literals. DR-DEVICE accepts as input the URL of a defeasible logic rule base. The facts for the rule program are contained in RDF documents, whose addresses are declared in the rule base. When the rule base is submitted, the designated facts are downloaded and the inference process commences. Rule conclusions are materialized inside DR-DEVICE and are then exported as an RDF document, including corresponding RDF/S definitions.

In our approach, reasoning services are wrapped by an agent interface, the *Reasoner*, allowing other IAs to contact them via ACL messages. The Reasoner can launch an associated reasoning engine that performs inference. In this work DR-

DEVICE was chosen, but our plans involve integrating a range of reasoning engines that use various logics. In essence, the Reasoner is a service and not an autonomous agent; the agent interface is provided in order to integrate into JADE.

The Reasoner constantly stands by for new requests (ACL messages with a "*REQUEST*" communication act). As soon as it gets a valid request, it launches DR-DEVICE that processes the input data (i.e. rule base) and returns an RDF document containing the results. Finally, the Reasoner returns the above result through an "*INFORM*" ACL message.

2.1. Trust Metric

There are various trust metrics [9]; our approach adopts a combination of *Sporas* (the most widely used) and *CR* (one of the most recently proposed), while using a decentralised reputation mechanism. Each agent keeps the references given from other agents and calculates the reputation value, according to the formula:

$$R_{i+1} = \frac{1}{\theta} \sum_{1}^{t} \Phi(R_i) R_{i+1}^{other} (W_{i+1} - E(W_{i+1})) \qquad (1)$$

$$\Phi(R) = 1 - \frac{1}{1 + e^{\frac{-(R-D)}{\sigma}}} \quad \text{and} \quad E(W_{i+1}) = \frac{R_t}{D}$$

W_i represents user *i*'s rating, t is the number of ratings the user has received, θ is a constant integer > 1, R^{other} is the reputation of the user giving the rating, D is the reputation value range and σ is the acceleration factor of damping function Φ. W_i is based on four coefficients: *Correctness* ($Corr_i$), *Response time* ($Resp_i$) and *Flexibility* ($Flex_i$); the latter refers to the Reasoner's flexibility in input parameters. The evaluation of the coefficients is based on user standards and their ratings vary from 1 to 10. The final rating value is the weighted sum of the coefficients (equation 2), where a_{i1}, a_{i2}, and a_{i3} are the respective weights and $nCorr_i$, $nResp_i$ and $nFlex_i$ are the normalized values for correctness, response time and flexibility, accordingly:

$$W_i = a_{i1} nCorr_i + a_{i2} nResp_i + a_{i3} nFlex_i \qquad (2)$$

New users start with R_i=0, $R_i \in [0, 3000]$, while $W_i \in [0.1, 1]$. As soon as the interaction ends, the Reasoner requests a rating. The other agent responds with a new message containing both its rating and its personal reputation and the Reasoner updates its reputation, according to equation 1.

2.2. Agent Communication Protocols

The protocol describes the allowed performatives in agent communication: the two main performatives in our approach are "*REQUEST*" that refers to the action

of submitting a request to perform a certain action, given certain preconditions, and "*INFORM*" that deals with returning the result of a previously submitted request to perform an action. A third, "*NOT-UNDERSTOOD*" performative is used if a non-appropriate message is received. The protocol is represented as a finite state machine with discrete states and transitions (e.g. Fig. 1).

3. Use Case: A Brokering Scenario

DR is useful in various applications, like brokering and bargaining [10], which are also extensively influenced by agent-based technology [11]. Here, a DR brokering scenario [12] is described that demonstrates the functionality of our approach.

3.1. Scenario Overview

A MAS is formed by three independent parties, represented by IAs: (a) the *customer* (called Carlo) is a potential renter who wishes to rent an apartment based on his requirements (e.g. location, floor) and preferences, (b) the *broker* possesses a list of available apartments, along with their specifications (stored as an RDF DB). His role is to match Carlo's requirements with the apartment features and propose suitable flats, (c) the *reasoner* is an independent, trusted third-party agent-based service, that uses DR-DEVICE in order to infer conclusions from a defeasible logic program and a set of facts and produces the results as an RDF file.

The broker does not wish to reveal his DB to customers, as it's one of his most valuable assets. However, the schema (RDF/S file containing properties and data types) *must* be exposed, so that customers can formulate their requirements. After inspecting the schema, Carlo expresses his requirements in defeasible logic (in the DR-DEVICE RuleML-like syntax) and his agent sends them to the broker, in order to retrieve all available apartments with the proper specifications.

The broker cannot directly process Carlo's defeasible logic requirements, as it internally uses a different logic, so a trusted third-party reasoning service is requested and is retrieved from the framework directory service or, alternatively, recommended by the customer. The broker sends Carlo's requirements to the Reasoner, along with the URI of the RDF DB containing all available flats, and stands by for the list of proper apartments. Alternatively, the broker could send the file itself instead of the URI. Also, not necessarily *all* the DB flats are available.

The Reasoner launches DR-DEVICE, which processes the above data and returns an RDF document, containing the apartments that fulfill all requirements. The result is sent back to the broker's agent and the latter, consecutively, sends it to Carlo's agent. Meanwhile, the broker can process the results in order to filter out some, using his own negotiating strategy. Eventually, the broker sends a rating to the Reasoner, updating its reputation accordingly.

Eventually, Carlo receives the list of appropriate flats and has to decide which one he prefers. However, Carlo does not want to send his preferences to the broker, because he is afraid that the broker might exploit that and not present him with the optimum choices. Thus, Carlo's agent sends the list of acceptable apartments and his preferences (again as a defeasible logic rule base) to the Reasoner. The latter calls DR-DEVICE, gets the single most appropriate flat and replies, proposing the best transaction. Carlo's agent could have an internal DR engine (e.g. [13]), but here we emphasize its ability to use any suitable external reasoning engine. The procedure ends, when Carlo sends his rating to the Reasoner and can now safely make the best choice based on his requirements and preferences.

3.2. Brokering Protocol

Although *FIPA* provides standardized protocols, none supports 1-1 automated brokering. Thus, a brokering protocol was implemented that encodes the allowed sequences of actions for the automation of the brokering process. The protocol (Fig. 1) is based on specific performatives that conform to the FIPA ACL specification. S_0 to S_6 represent states of a brokering trade and E is the final state. Predicates *Send* and *Receive* represent interactions that cause state transitions. The sequence of transitions for the customer is $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow E$, while, for the broker it is $S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow E$. In case an agent receives a wrong performative, it sends back a "*NOT-UNDERSTOOD*" message and the interaction is repeated.

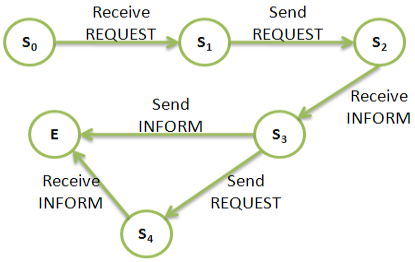

Fig. 1. Agent brokering communication protocol.

4. Conclusions and Future Work

This paper argued that reasoning interoperability among SW agents can be achieved via trusted, third-party reasoning services to provide reasoning capabilities in a variety of logic and rule-based formalisms. Towards this direction, a JADE MAS was presented, whose main component is a DR service implemented as an agent. Also, trust metrics have been studied and a reputation mechanism for

the reasoning service has been embedded. Finally, the paper presented a use case brokering trade scenario that illustrates the usability of the proposed approach.

A related effort is the *Rule Responder* (*RR*) [14] project that builds a service-oriented methodology and a rule-based middleware for interchanging rules in virtual organizations. *RR* exhibits the interoperation of distributed platform-specific rule execution environments via Reaction RuleML as a platform-independent rule interchange format. The main differences with our approach are: (a) *RR* assumes a unique rule interchange language, while in our approach this is not necessary, (b) in *RR* rule engines are incorporated *inside* agents, whereas we deploy them as independent service-agents, and (c) our framework is totally FIPA-compliant, whereas RR introduces a new RuleML-based agent communication interface.

Concerning future directions for our approach, we would like to verify our architecture's capability to adapt to a variety of additional scenarios, like negotiation. Another goal is to integrate a variety of reasoning engines, in order to test and evolve the reasoning interoperating capabilities of our framework.

References

1. Berners-Lee T, et al. (2001) The Semantic Web. Scientific American, 284(5):34-43
2. Hendler J (2001) Agents and the Semantic Web. IEEE Intelligent Systems, 16(2):30-37
3. Genesereth M, Fikes R (1992) Knowledge Interchange Format Version 3.0 Reference Manual. Technical Report, Logic Group, Computer Science Dept., Stanford University
4. Boley H (2006) The RuleML Family of Web Rule Languages. Proc. PPSWR, Vol. 4187, Springer, pp. 1-17
5. Kifer M, (2008) Rule Interchange Format: The Framework. Proc. 2nd International Conference on Web Reasoning and Rule Systems, Karlsruhe, Germany, pp. 1-11
6. Wagner G et al. (2005) R2ML: A General Approach for Marking up Rules. Dagstuhl Seminar Proceedings 05371, Principles and Practices of Semantic Web Reasoning
7. Bassiliades N, Antoniou G, Vlahavas I (2006) A Defeasible Logic Reasoner for the Semantic Web. IJSWIS, 2(1):1-41
8. Nute D (1987) Defeasible Reasoning. Proc. 20th International Conference on Systems Science. IEEE Press, pp. 470-477
9. Macarthur K. (2007) Trust and Reputation in Multi-Agent Systems. Proc. AAMAS'08, Estoril, Portugal, May 12-16
10. Benjamins R, Wielinga B, Wielemaker J, Fensel D (1999) An Intelligent Agent for Brokering Problem-Solving Knowledge. IWANN, (2):693-705
11. Skylogiannis T, Antoniou G, Bassiliades N, Governatori G, Bikakis A (2007) DR-NEGOTIATE - A System for Automated Agent Negotiation with Defeasible Logic-based Strategies. DKE, 63(2):362-380
12. Antoniou G, Harmelen F van (2004) A Semantic Web Primer. MIT Press
13. Antoniou G, Skylogiannis T, Bikakis A, Bassiliades N (2005) DR-BROKERING - A Defeasible Logic-Based System for Semantic Brokering. Proc. IEEE International Conference on E-Technology, E-Commerce and E-Service, IEEE, pp. 414-417
14. Paschke A, Boley H, Kozlenkov A, Craig B (2007) Rule responder: RuleML-based Agents for Distributed Collaboration on the Pragmatic Web. Proc. 2nd International Conference on Pragmatic Web. ACM, (280):17-28, Tilburg, The Netherlands