Matchmaking Using an Instance Store: Some Preliminary Results

Lei Li and Ian Horrocks
University of Manchester, Manchester, UK
Email: {lil, horrocks}@cs.man.ac.uk

Abstract

An important objective of the Semantic Web is to make Electronic Commerce interactions more flexible and automated. To achieve this, standardization of ontologies, message content and message protocols will be necessary.

In this paper we investigate how Semantic and Web Services technologies can be used to support service advertisement and discovery in e-commerce. In particular, we present some preliminary results of a service matchmaking prototype which uses an *instance store* to retrieve ontology based individual descriptions.

1 Motivation

The Semantic Web requires that data be not only machine readable (just like the Web nowadays does), but also that it be machine understandable. It should facilitate the realisation of a whole range of tools and applications that are difficult to handle in the framework of the current web. Examples include knowledge-repositories, search agents, information parsers, etc. Ontologies will play an important role in the Semantic Web by providing vocabularies with explicitly defined and machine understandable meaning [8].

One important Semantic Web application area is e-commerce. In particular, a great deal of attention has been focused on semantic *web services*, the aim of which is to make services more accessible to automated agents. Here, ontologies can be used to describe services so that agents (both human and automated) can advertise and discover them according to a semantic specification of functionality (as well as other parameters such as cost, security, etc.) [11].

If applications are to exchange semantic information, they will need to use common ontologies. In this paper, we present a case study of an e-commerce application in which a predefined service ontology is used to provide the vocabulary for service descriptions. These descriptions are used in a matchmaking prototype, i.e., a repository where agents can advertise and search for services that match some semantic description.

With the possibilities opened up by e-commerce, the number of potential service advertisements could be huge, and millions of them might need to be examined in order to find relevant ones. However, loading and reasoning over such large amounts of ontology based information seems likely to be an intractable task for current Abox reasoners. To resolve this problem, a simple DL *instance store* has been developed by the IMG group at the University of Manchester. It provides a weaker Abox assertion/query language able to support applications where individual objects are only described in terms of concepts that they instantiate, and are subsequently retrieved using concept description queries [2].

In our prototype, we used the JADE [6] agent platform to simulate advertising and querying agents, and we used the DL *instance store* in order to manage all the service advertisements and service requests. The Racer DL reasoner [9] is used behind the *instance store* to compute semantic matches between service advertisements and service requests. We carried out a performance analysis using this prototype in order to discover if the approach is likely to be feasible in large scale web applications.

2 Background

2.1 Description Logic

Description Logics are a well-known family of knowledge representation formalisms. They are based on the notion of concepts (unary predicates, classes) and roles (binary relations), and are mainly characterised by constructors that allow complex concepts and roles to be built from atomic ones [5]. A DL reasoner can be used, e.g., to check if concepts are consistent, or whether two concepts subsume each other [4]. We assume the reader to be familiar with DLs—see [1] for a detailed discussion of DLs.

2.2 Instance Store

A web service repository may have to cope with a very large number of advertisements, each of which is treated as an instance of an associated service description. It seems unlikely that traditional Abox reasoning would be able to cope with this number of individuals [10].

In this application, however, there are no axioms asserting relationships between pairs of individuals, an it is sufficient to use the weaker functionality provided by an instance store. The instance store is implemented using a conventional database and a reasoner. The database stores individuals and their descriptions, as well as the KB classes that individuals instantiate (this is computed when an individual is added to the instance store). By using a combination of queries against the database and subsumption/classification requests to the reasoner, the instance store can answer retrieval[1] and realisation[2] queries which would otherwise require full Abox reasoning.

[1]Which individuals are instances of concept description C?

[2]What are the most specific concept names in the TBox that an individual i is an instance of?

A complete description of the instance store is referred to [2].

3 Matchmaking Using Instance Store

Service description ontologies will have an important role to play in our work, so we have designed a domain-specific sample ontology about the sales of computers in order to achieve matching at the semantic level between various parties. For the purpose of clarity and compactness, in this paper we will use the standard DL abstract syntax.

In our ontology, we use *ServiceProfile* as a common superclass; other categorized service are expressed as specialised subclasses of *ServiceProfile*, e.g.:

$$\begin{array}{rcl} ServiceProfile & \sqsubseteq & \top \\ SmallSerivce & \equiv & ServiceProfile \sqcap (PC \sqcap \leq_{200} unitQuantity) \\ AMDOnlyService & \equiv & ServiceProfile \sqcap (PC \sqcap \forall hasCpu.AMDCpu) \\ & & \\ SmallAMDService & \sqsubseteq & SmallService \sqcap AMDOnlyService \\ \ldots & & \end{array}$$

Suppose that we want to define an advertisement as an instance of the following service description: the provider is a large service provider, them items are PCs and the processor brand must be Intel and the unit price must be less than 500. In DL notation, this advertisement `Advert1` can be written as:

$$\begin{array}{rcl} Advert1 & \in & LargeService \sqcap IntelOnlyService \sqcap \\ & & ServiceProfile \sqcap (PC \sqcap <_{500} unitPrice) \end{array}$$

Matchmaking is defined as a process that requires a repository host to take a query as input, and to return all advertisements which may potentially satisfy the requirements specified in the input query. I.e., for α a set of advertisements and Q a query, the set of *compatible* advertisements $matches(Q)$ is defined as: $matches(Q) = \{A \in \alpha | compatible(A, Q)$, where $compatible(D1, D2)$ iff $D2 \sqsubseteq D1$ (w.r.t. the service ontology).

4 Implementation and Evaluation

We have implemented a prototype system for matchmaking using the DL textitinstance store. We chose JADE as the agent platform, the goal of JADE being to simplify the development of multi-agent systems while ensuring standard compliance through a comprehensive set of system services and agents in compliance with FIPA [3] specifications. Three kinds of agents have been implemented—HostAgent, Seeker and Advertiser. At the beginning of the matchmaking process, the HostAgent initializes the instance store with the service ontology described in Section 3. When it receives an advertisement, the HostAgent asserts it and stores all the relevant information in the instance store. When it receives a request, the HostAgent uses the instance store system to retrieve all the compatible advertisements. The matched result for this request is then returned to the seeker agent. See [7] for further implementation details.

We used the prototype implementation to carry out some simple experiments designed to test the system's performance in an agent based e-commerce scenario. The experiment used datasets of between 200 and 20,000 (artificially generated) advertisements, and recorded the time taken by the instance store to find matched advertisements in response to a given request[3]. To make the performance test more interesting, we also tested Racer's Abox reasoning using the same datasets. Our results showed in Figure 1 that:

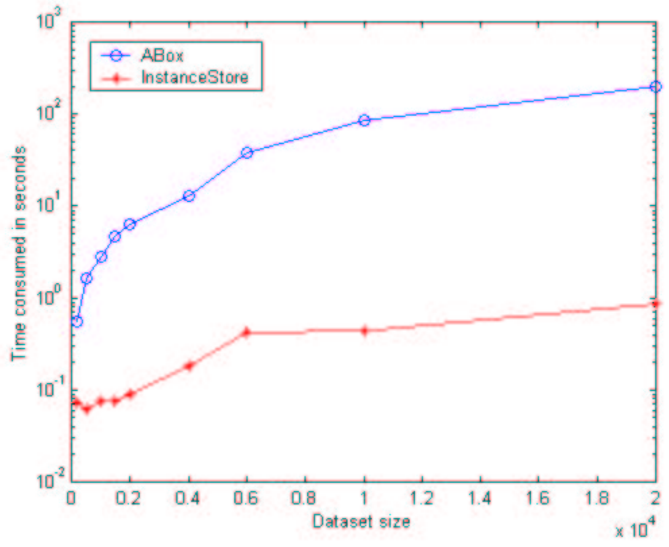

Figure 1: Classification time comparison

Instance store: the retrieval time required to respond to a matching request is always less than one second. This is an encouraging result considering that the largest dataset size is twenty thousand advertisements. Moreover, there is relatively little change in performance as the number of advertisements is increased.

Racer Abox: the retrieval time is much worse compared to the instance store: it takes almost 200 seconds with a dataset of twenty thousand advertisements. Moreover, query time increases significantly with increased numbers of advertisements.

[3]The advertisements are generated as the instances of some random generated concept using the combination of the primitive concepts in the taxnomy. The requests are treated as some predefined concept using the combination of the primitive concepts.

5 Conclusions and Future work

Ontology based service descriptions can benefit service discovery in e-commerce. By representing the semantics of service descriptions, the matchmaker enables the behaviour of an intelligent agent to approach more closely that of a human user trying to locate suitable web services. In order to demonstrate the feasibility of this approach, we have implemented a prototype matchmaker which uses an instance store to match service advertisements and requests based on the semantics of ontology based service descriptions.

In this paper we have presented some preliminary results from our prototype. These results suggest that, when using the instance store, matchmaking based on DL reasoning could be used to search for web services on a large scale.

References

[1] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook.* 2002.

[2] Sean Bechhofer, Ian Horrocks, and Daniele Turi. Instance store: Database support for reasoning over individuals. `http://instancestore.man.ac.uk`.

[3] FIPA portal, http://www.fipa.org/.

[4] I. Horrocks and P. F. Patel-Schneider. Comparing subsumption optimizations. In *Collected Papers from the International Description Logics Workshop*, 1998.

[5] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR'99)*, 1999.

[6] JADE portal, http://jade.cselt.it/.

[7] Lei Li and Ian Horrocks. A software framework for matchmaking based on semantic web technology. In *Proceedings of the Twelfth International World Wide Web Conference (WWW 2003)*, 2003.

[8] D. L. McGuinness. Ontological issues for knowledge-enhanced search. In *Proc. of FOIS*, Frontiers in Artificial Intelligence and Applications. IOS-press, 1998.

[9] RACER portal, http://www.fh-wedel.de/˜mo/racer/.

[10] S. Tessaris. *Questions and answers: reasoning and querying in Description Logic*. PhD thesis, University of Manchester, 2001.

[11] D. Trastour, C. Bartolini, and C. Preist. Semantic web support for the business-to-business e-commerce lifecycle. In *Proceedings of the Eleventh International World Wide Web Conference (WWW 2002)*, 2002.