Description Logics for Matchmaking of Services.

Javier González-Castillo, David Trastour*and Claudio Bartolini
HP-Labs, Bristol, UK
Email: javgon@hplb.hpl.hp.com, david_trastour@hp.com,
claudio_bartolini@hp.com

Abstract

Matchmaking is an important aspect of E-Commerce interactions. The current trend in B2B E-Commerce automation is towards complex interactions for service provision. In this context, matchmaking services require rich and flexible metadata as well as matching algorithms. The Semantic Web initiative at W3C is gaining momentum and generating suitable technologies and tools to cover both the metadata and the algorithmic aspects. In this paper we describe our experience in building a matchmaking prototype. We choose to base our prototype on a Description Logic (DL) reasoner, operating on service descriptions in DAML+OIL. We report on our investigation of DAML+OIL to express service descriptions and on our experience on existing DL reasoners, in particular assessing RACER and FACT.

1 Introduction

The automation side of E-Commerce transactions brings many advantages to businesses in dealing with their partners, customers, and suppliers. The increased efficiency and fewer errors in computations make possible higher throughput and further reach, and therefore open up the possibility of interacting with a far greater number of potential counterparts. But with the enlarged possibilities comes the problem of having to select the best among the multitude of available counterparts. Such selection must happen based on various aspects of the business offers that providers make available and requestors seek for. Matchmaking is the process of pruning the space of possible matches among compatible offers and requests.

*Please consider David Trastour as the main contact.

There are two factors that play in making matchmaking in B2B E-Commerce a difficult problem. On one hand, service provision interactions evolve to be ever more complex. This requires that the language for service descriptions for matchmaking be expressive enough to deal with this complexity. On the other hand the sheer number of potential solutions heavily constrains the efficiency that is achievable. This, united with a requirement for accuracy in reporting matching offers and requests, makes the problem barely tractable with traditional techniques.

Because Semantic Web technologies promise to transform the information on the web from human-readable to machine-understandable [14], we think that the application of these technologies may be valuable to our aim. In particular, we have studied DAML+OIL as we believe that a subset of it could be used to describe service parameters. Because DAML+OIL is heavily influenced by Description Logics, it seems natural to use a DL reasoner as the heart of the engine that calculates the matches.

The remainder of this paper is structured as follows. Section 2 describes matchmaking in detail. In section 3 we analyze how DL could be a solution for matchmaking and we describe a matching algorithm based on the subsumption tree given by a DL reasoner. In section 4 we describe our experience with different DL reasoners and list some requirements we would like to see for future DL reasoners. Section 5 talks about future work and we conclude in section 6.

2 Matchmaker

With the proliferation of offers comes the problem of finding and selecting potential counterparts for service provision/consumption. The sole presence of many potential buyers and sellers on the web is not a sufficient condition for them doing business together. Through the mediation of the matchmaker, which matches service offers with service requests, potential counterparts will be able to find each other.

2.1 Service Description Language

Service description is a very broad term subject to different interpretations. For example, WSDL (Web Service Description Language) [8] descriptions focus on the behavioral aspects of a service. UDDI (Universal Descripition, Discovery and integration of Business for the Web) [5] descriptions are based on three different types of information: contacting details – white pages –, classification with respect to a certain taxonomy – yellow pages –, and technical information – green pages.

The purpose of our work is to embrace and extend Web Services descriptions,

taking a more general approach while providing the expressiveness and flexibility that we require. Our approach is based on expressing service descriptions through a set of complex parameters. These parameters are used to express a variety of aspects of the service and the entities involved. Our framework must be flexible enough to accommodate descriptions with various levels of complexity, from the simple sale of a good to a complex business interaction.

2.1.1 Requirements

Previous investigations [17] on the application of RDF/RDFS [14, 6] to service description in the context of matchmaking lead us to he following requirements:

- **High degree of flexibility and expressiveness.** The advertiser must have total freedom to compose the service description. Different advertisers will want to describe their services with different degrees of complexity and completeness, and our language must be adaptable to these needs. An advertisement may be very descriptive in some points, but leave others less specified and open for negotiation a posteriori. Therefore, ability to express semi-structured data is required.

- **Support for Types and Subsumption.** We do not want to restrict matching to be based on simple string comparison. A type system with subsumption relationships is required, so more complex matches can be provided based in these relationships.

- **Support for Datatypes.** Attributes such as quantities, prices, or dates will be part of the service descriptions. The best way to express and compare this information is by means of datatypes. As a starting point, we will deal with datatypes such as `real`, `date`, `string` etc.

- **Express Restrictions and Constrains.** Whether it is an offer or a request, it is often the case that what is expressed is not a single instance of a service but rather a conceptual definition of the acceptable instances. A natural way of describing this is by expressing constraints over the parameters of the service.

- **Semantic level of Agreement.** In order to compare descriptions, they need to share the same semantics.

- **Appropriate syntax for the Web.** The matchmaker must be compatible with Web technologies and the information must be in a format appropriate for the Web.

2.1.2 DAML Approach

DAML+OIL is one of the most promising technology of the Semantic Web activity. This ontology mark-up language for web resources developed by DARPA provides a richer set of modeling primitives than other ontology languages such as RDF/RDFS. In its last specification [18] it has been extended with arbitrary datatypes from XML Schema [4].

To fulfill our requirements, we are proposing to represent the concepts in a service description as DAML+OIL classes. The service description is defined as the boolean combination of a set of restrictions over datatype and abstract properties. These restrictions are expressed either through DAML+OIL restrictions or XML Schema restrictions. It is worth noting that service description ontologies and domain-specific ontologies also have an important role to play in order to achieve the semantic level of agreement between the various parties. The example service description ontology we have developed uses the class `srcv:ServiceDescription` to denote the root of a service description.

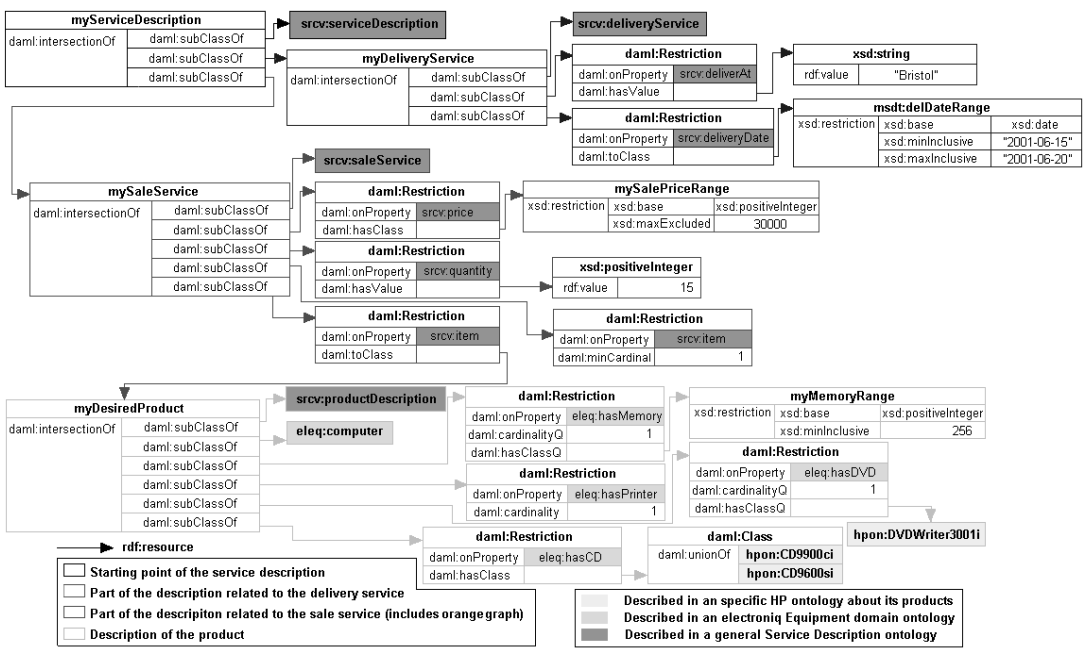

Figure 1: Example: Description of a service

Let us consider an example of a composite service of sale and delivery of computers. For the sole purpose of this example, we have defined a service description ontology and some electronic equipment related ontologies. We want to emphasis on how DAML+OIL is used to describe services requests and offers. Figure 1 represents the service of sale of 15 items. These items must be of type

`eleq:computer` and must satisfy the following constraints: has at least 256 MB of RAM memory; has a printer – any make –; has a DVD unit, model HP DVDWriter 3001i; and has a CD unit, HP-CD9900ci or HP-CD9600si. There is an additional restriction on this service of sale stating that the price must be less than 30,000. The service of delivery has the following constraints: the goods must be addressed to "Bristol" any day between 15-06-2001 and 20-06-2001 included.

In order to construct this description we make use of different classes and properties. Some of them have been previously defined in some domain specific ontologies written by third parties - boxes filled with different colors -, while others such as `myServiceDescription`, `mySaleService`, `myDeliveryService` and `myDesiredProduct` are defined here in terms of restrictions over properties. All boxes in the figure named `daml:Restriction` represent these restrictions. Depending on the slots inside these boxes, we can distingish different types of restrictions such as: existential restrictions, value restrictions or cardinality restrictions.

2.2 Concept of match

Different approaches to matching can be taken. Existing solutions like UDDI or ebXML[1] (Electronic Business XML) manage to give accurate results at the expense of expressiveness by having a rigid format for descriptions and by restricting the query mechanism. Based on real-life examples like yellow pages directories, advertisement newspapers or bulletin boards, we would rather be able to compare descriptions with different levels of specificity and complexity than use an approach based on exact matching. For instance a *general* description for the sale of PCs, without any restrictions, should match the above example. More *specific* descriptions should also be matched. Finally, descriptions that are neither more specific or more general but that describe services that would be compatible with our example should also match.

Definition 1 *A service description is a self-consistent collection of restrictions over named properties of a service.*

Definition 2 *A service description D1 is a match for a service description D2 if there is no contradiction between all of the restrictions in D1 and D2.*

In the following section, we look at how DL reasoners can help us find matches among DAML+OIL based service descriptions.

[1] Suite of specifications that enables enterprises to conduct business over the Internet.

3 Description Logics

Description Logics are a family of knowledge representation formalisms. They are based on the notion of concepts and roles, and are mainly characterised by constructors that allow complex concepts and roles to be built from atomic ones [11]. The main benefit from these knowledge languages is that sound and complete algorithms for the subsumption and satisfiability problems often exist. A DL reasoner solves the problems of equivalence, satisfiability and subsumption.

3.1 DL and DAML+OIL

Because DAML+OIL has been influenced by DL, it appears natural to apply DL techniques to classsify our service descriptions.

As we will see in the following section, none of the DL variants for which there exists an implementation of a reasoner possesses enough expressiveness to deal with the whole set of constructors that form the DAML+OIL language. If we want to adopt a DL solution for implementing the matchmaker, we must restrict the descriptions to a subset of DAML+OIL.

At the moment, the most advanced available reasoners are for the $\mathcal{SHIQ}$ DL. This DL supports most of the DAML+OIL language, but its main drawback is that it cannot deal with individuals or datatypes in the definition of concepts. From our list of requirements this is too restrictive.

$\mathcal{SHOQ}$(D) is more expressive than $\mathcal{SHIQ}$ as it adds individuals and datatypes support – even though it does not allow for inverse role[2]. Not having the inverse role property does not cause us any major concern while individuals and concrete datatypes are quite essential to our application. A complete algorithm for solving the subsumption and satisfiability problem for $\mathcal{SHOQ}$(D) exists [12], but we do not know of any implementation available.

3.2 DL for Matchmaking

In this section we are describing the functionalities of a matchmaking service and a DL based matching algorithm needed to provide them.

3.2.1 Matchmaker functionalities

Our matchmaking service provides three basic functionalities [17]:

Advertising is the act of publishing a service description, or advertisement, to a matchmaking service. Before an advertisement is included in the

[2]Reasoning with both concrete domains or individuals plus inverse roles is known to be difficult and/or highly intractable [12].

knowledge base of the matchmaker, the satisfiability of all its concepts must checked because the realization of a non satisfiable service is not possible. When accepted, an advertisement becomes a set of new concepts within the subsumption tree. One of these concepts, the one under the *serviceDescription* branch, represents the whole advertisement.

Querying is similar to advertising except that the description submitted to the matchmaker is not persistent. The algorithm in the next section let us calculate the matches with a DL reasoner.

Browsing allows parties to find out about published advertisements. Browsing parties can make use of this information to tune the advertisement or queries that they submit in turn, so as to maximize the likelihood of matching. Browsing is based on navigating the subsumption tree through the branches provided by our service description ontology.

3.2.2 Matchmaking Algorithm

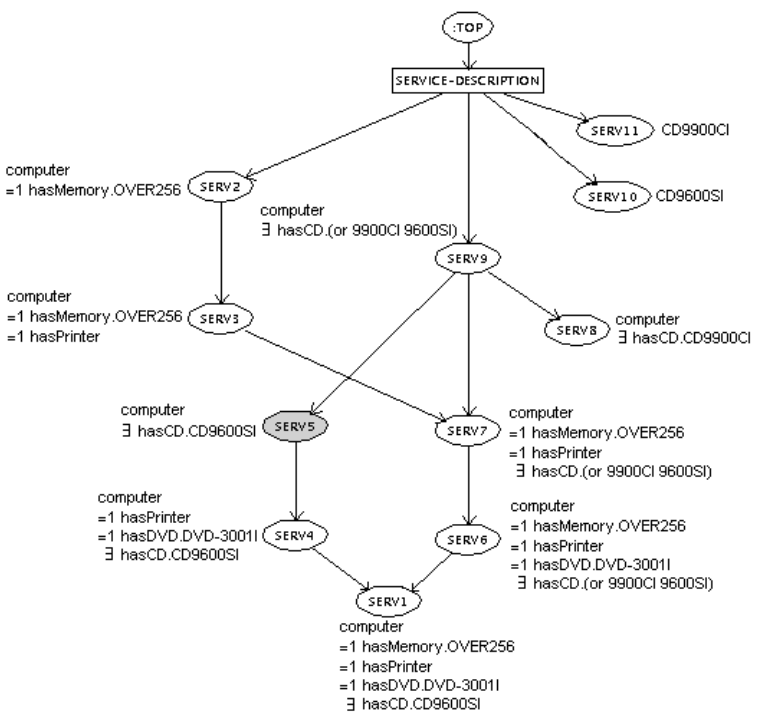

Figure 2: Service description branch of the subsumption tree

Definition 3 *The matches for service description S are:*

- *equivalent concepts to S;*
- *sub-concepts of S;*
- *super-concepts of S that are subsumed by the serviceDescription concept;*
- *sub-concepts of any direct super-concept of S whose intersection with S is satisfiable.*

The algorithm is a translation in DL terms of the ideas exposed in the previous section.

To understand the algorithm in more detail, we are applying it to the example depicted in Figure 2. This figure shows the *serviceDescription* branch of the subsumption tree in the matchmaker at the moment of the query. Nine advertisements of sale and delivery of computers and two of sale of CD units have been published. We are considering a party interested in finding a computer which has a CD unit CD9600Si. Her query is denoted as SERV5 in the figure (filled node).

We evaluate sequentially the four propositions of the algorithm. In our example, there is no equivalent concept to SERV5. SERV4 and SERV1 are subconcepts of SERV5 and as such are marked as matches. The third step is to look for super-concepts of SERV5 – up to the *serviceDescription* node. Hence, SERV9 is marked as a match. Finally, the last step of the algorithm gives us the nodes SERV6 and SERV7. While these nodes are neither super or sub concepts, they are compatible to the SERV5 query[3], in that the restrictions over the properties that appear in them and SERV5 are not inconsistent.

The problem of presenting results back to the user in a way that make sense to her (i.e. any ordering based on preferences) is beyond the scope of our current work.

4 Practical Approach

In this section we report on our experience on existing DL reasoners, in particular assessing RACER and FACT. We also list a set of requirements for future DL reasoner suitable for our application.

4.1 Experiences with existing DL Reasoners

4.1.1 FaCT Reasoner

The FaCT [13] system is a DL classifier being developed by Ian Horrocks from the Department of Computer Science at the University of Manchester. It includes

[3]Even though SERV8 is a sub-concept of SERV9 – which is a match –, it is not given as result because it contradicts the SERV5 query.

two reasoners for TBoxes, one of them for the $\mathcal{SHIQ}$ logic. Therefore, it cannot deal with individuals or concrete datatype domains, and a description such as the one in Figure 1 can not be processed with it.

To cope with the limitation of $\mathcal{SHIQ}$, we tried to model nominals, datatypes, and datatype values as atomic concepts but this can lead to incorrect inferences [12], not to mention the need to model one atomic concept for each integer.

DAML+OIL uses namespaces and import statements to provide extensibility and to deal with the distributed nature of the Web. The support in the reasoner for multiple interconnected TBoxes would solve this problem as we would model each DAML+OIL ontology in a different TBox. Because FaCT does not support multiple TBoxes we are using fully qualified names in a single TBox.

Moreover, the knowledge base of the matchmaker will change over time by addition of new advertisements as well as deletion or modification of existing ones. FaCT deals with the addition of new classes over time, even after classification has been done, but doesn't provide a mechanism for removing classes in the classification. This is a requirement for our application.

One of the main benefits of this system is its CORBA interface [3] that makes the reasoner available as a service for other applications to use. It also provides XML syntax for the definition of ontologies. To load our descriptions in the reasoner, we are translating DAML+OIL descriptions to the FaCT XML syntax.

4.1.2 RACER Reasoner

RACER [9, 10] is the first reasoner for TBox and ABox for the $\mathcal{SHIQ}$ logic. It is developed at the Computer Science Department of the University of Hamburg.

Like FaCT, it only provides part of the expressiveness that we need for our application. It is able to deal with multiple TBoxes, but they are not interconnected. It does not let us define a concept in a TBox in terms of concepts or roles from other Tboxes.

RACER does not provide support for a dynamic knowledge base as it is not possible to add or remove concepts once the classification has been done.

Another interesting feature of RACER is its ability to reason about ABoxes. With our approach to matchmaking this capability is not strictly necessary, as we only need to reason about concepts, for which TBoxes provide the necessary abstraction. However, the ability to reason about ABoxes may prove useful when extending our framework to cover phases of E-Commerce transaction beyond matchmaking. For example, an agreement struck between two parties following matchmaking and automated negotiation [2], needs full instantiation of the parameters that originally appeared in the service descriptions. Support for ABoxes would enable compliance check of the agreement with the negotiation proposals and with the original service descriptions in turn.

RACER provides a Java API and allows access to the reasoner remotely.

4.2 Requirements for a DL reasoner for matchmaking

From our experience, we have gathered the following requirements for a DL reasoner that would satisfy our needs:

- $\mathcal{SHOQ}(\mathbf{D})$ is the minimum expressiveness required;
- **Dynamic.** Advertisements will be added, removed and modified and the concepts within the knowledge base will need to be re-classified.
- **Ability to deal with multiple interconnected Tboxes.** We want to use different ontologies, and define concepts based on other external concepts and roles.
- **Scalability.** The reasoner needs to be able to cope with large amounts of information in an efficient way.
- **Persistency.** Storage of the advertisements is needed. The reasoner needs to be integrated with some form of persistent store in a way that maintains data consistency.
- **Support for DAML+OIL syntax** would avoid unnecessary translations.

5 Future Work

Our two prototype implementations of a matchmaker are fairly similar in terms of functionalities and are both incomplete. To go further into the development, we are lacking a DL reasoner with the properties mentioned above. The main requirement would be the support for the right level of expressiveness: $\mathcal{SHOQ}$(D).

On the service description side, we realize that the model we are proposing restricts the description of the service to a set of parameters. While this approach fits well with simple services like catalog-based solutions for the sale of goods, we recognize the need for a behavioural description for complex services. While all the examples of this paper only exposed buyer-seller relationships, we need to envisage a world where parties want to interact though complex business processes. The matchmaking of potential counterparts would then need to consider not only the service parameters, but also the compatibility between the various roles and behaviours. We want to include this work with another activity we are carring out on cooperative business processes [15].

Recently the DAML community has announced and release a first version of DAML-S, the Web Service Mark-up Language [7]. DAML-S is a Web service

ontology which will allow software agents to discover, invoke, compose and monitor the execution of Web Services. To validate our ideas we have developed a primitive service description ontology. Our work could only benefit from using a full-fledged service description ontology. We will try to leverage from DAML-S as much as we can.

We are tracking what the Semantic Web community is producing in terms of tools but more specifically persistent stores for RDF and DAML. In particular we find the work on RQL [1] very promising. We need to envisage how to integrate DL reasoners with a persistent store.

6 Conclusions

Our experience in prototyping a DL based matchmaking service made us realize that there is a gap between what standard technologies for E-Commerce provide today and what could be achieved through the use of Semantic Web technologies. We believe that in the near future automated matchmaking and negotiation will achieve results at a level of complexity far beyond what is possible today. In particular the use DAML+OIL and of the evolution of DL reasoners like FaCT or RACER will play a primary role in making that happen.

References

[1] V. Alexaki; et al. *The ICS-FORTH RDFSuite: Managing Voluminous RDF Description Bases, Proceedings of the Second International Workshop on the Semantic Web.* SemWeb'2001. May 2001.

[2] C. Bartolini and C. Preist *A Framework for Automated Negotiation.* 2001. HP Labs Technical Report.

[3] S. Bechhofer, I. Horrocks and S. Tessaris. *CORBA interface for a DL Classifier.* March 1999.

[4] P.V. Biron, A. Malhotra. *XML Schema Part 2: Datatypes.* W3C Recommendation 02 May 2001.

[5] T. Boubez; et al. *UDDI Data Structure Reference V1.0.* September 2000.

[6] D. Brickley and R.V. Guha. *Resource Description Framework (RDF) Schema Specification 1.0.* W3C Candidate Recommendation 27 March 2000.

[7] M. Burstein et al. *DAML-S: Semantic Markup for Web Services.* Part of the DAML-S Draft Release (May 2001).

[8] E. Christensen, F. Curbera, G. Meredith and S. Weerawarana. *Web Services Description Langauge (WSDL) 1.1.* January 2001.

[9] V. Haarslev and R. *Möller. RACER User's Guide and Reference Manual Version 1.5.6.* April 2001.

[10] V. Haarslev and R. *Möller. RACER System Description.* To appear in: International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy.

[11] I. Horrocks, U.Sattler and S. Tobies. *Practical reasoning for expressive description logics.*In H. Ganzinger, D. McAllester, and A. Voronkov, editors, Proceedings of LPAR'99, vol. 1705 of LNAI, pages 161–180. Springer, 1999.

[12] I. Horrocks. *Ontology Reasoning for the Semantic Web.* Network Inference 2001.

[13] I. Horrocks. *FaCT Reference Manual Version 1.6.* August 1998.

[14] O. Lassila and R. Swick. *Resource Description Framework (RDF) Model and Syntax specification.* W3C Recommendation 22 February 1999.

[15] G. Picinelli and L. Mokrushin. *Dynamic Service Aggregation in Electronic Marketplaces.* 2001. HP Labs Technical Report.

[16] U. Sattler. *A concept language extended with different kinds of transitive roles.* In 20. Deutsche Jahrestagung *für* KI, LNAI 1137.

[17] D. Trastour, C. Bartolini and J. Gonzalez. *A semantic Web Approach to Service Description for Matchmaking of Services.* Semantic Web Workshop Symposium 2001. To appear.

[18] F. van Harmelen, P.F. Patel-Schneider and I. Horrocks *Reference description of the DAML+OIL (March 2001) ontology markup language.* March 2001.