Description Logics Approach to Semantic Matching of Web Services

S. Colucci (1), T. Di Noia (1), E. Di Sciascio (1), F. M. Donini (2), M. Mongiello (1)
(1) Politecnico di Bari Via Re David, 200 – 70125 BARI, Italy
(2) Universitá della Tuscia,via S. Carlo 32, 01100 Viterbo, Italy
e-mail{s.colucci,t.dinoia,disciascio,mongiello}@poliba.it; donini@unitus.it

Abstract

As more resources and services become available on the Web, there is a growing need for infrastructures that, based on advertised descriptions, semantically match in a peer-to-peer way providers with requesters of web services. We address the problem of matchmaking of web services from a knowledge representation perspective. Based on our approach we propose match categorization in terms of exact match, potential match –when request and offer though not identical are compatible– and partial match –when one or more inconsistency is present– and rank of matches within categories. Then we report on our implementation of the proposed matchmaking framework in a prototype system.

1 Introduction

The advent of the Internet and of its most noticeable offspring, the World Wide Web, has initially led to an explosion of information available on-line. The lack of infrastructures to help in accurately locate and keep up to date the huge amount of information and resources at hand, has resulted, in a few years, in an information overload, basically because of the inherent unstructured nature of web data.

The Semantic Web [1] initiative has begun, slowly yet steadily, to revolutionize the way information is provided on the Internet. The basic idea is to structure information with the aid of markup languages, based on the XML language, such as RDF [21] and DAML+OIL [18]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of arbitrary domain ontologies, and aim at increasing openness and interoperability in the web environment. Building on the Semantic Web foundations, also a flourishing of languages for the description of web services has been recently witnessed, struggling to become accepted de-facto standards, such as WSDL [8] and DAML-S [24].

Widespread availability of resources and services enables — among other advantages — the interaction with a number of potential counterparts. The bottleneck is that it is difficult finding matches, possibly the best ones, between parties. *Matchmaking* is the process of searching the space of possible matches between demands and supplies, finding the best available ones[12, 20, 13, 23, 25, 24, 26, 14, 19, 27, 16, 7, 9]. Demands and supplies are here meant to represent web services, information, tangible or intangible goods, practically finding for any request an appropriate offer, or vice versa. Matchmaking is quite different from simply finding, given a demand a perfectly matching supply (or vice versa). Instead, it includes finding all those supplies that can *to some extent* fulfill a demand, and eventually propose the best ones. So the scenario, which is basically envisaged by the Semantic Web initiative, is one where peer entities may propose their goods and services and dynamically deal with counteroffers or further specifications, through the mediation of a matchmaking infrastructure.

A matchmaking infrastructure should then receive and store advertisement descriptions by both demanders and suppliers, and as dynamically new demands or supplies are submitted find most satisfying matches and return them. The infrastructure has to treat in a uniform way suppliers and demanders, and base the matches on common, extensible, ontologies for describing both supplies and demands [11].

Knowledge representation — in particular Description Logics (DL) — can deal with a uniform treatment of knowledge from suppliers and demanders, by modelling both as generic concepts to be matched.

In fact, the logical approach — which DL are based upon — allows for an open-world assumption. Incomplete information is allowed (and can be filled after a selection of possible matches), and absence of information can be distinguished from negative information, allowing to discard offers/requests without the necessary properties, and to ask for missing information in the potential matches. The importance of ranking can not be underestimated, as it is of extreme importance for a practical use of the approach. The key questions that have to be answered in a dynamic frame-

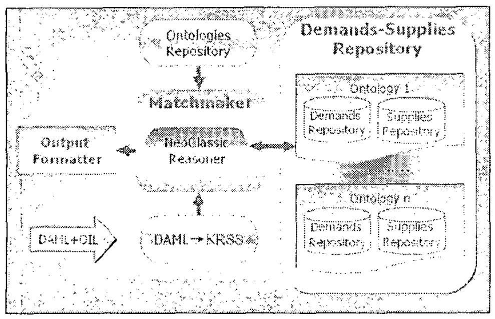

Figure 1. The architecture of the engine

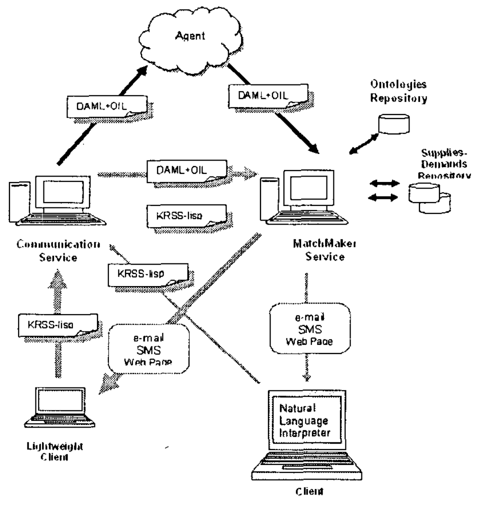

Figure 2. The architecture of the matchmaking infrastructure

work are how far is a given demand (supply) from a potential counterpart? And which are the requirements that would eventually fulfill it? Such questions have to be answered relying on publicly available algorithms, to enforce trust,and prevent arising of doubts on fairness of the proposals returned by the matchmaking facilitator.

In this paper, we will concentrate on descriptions, regardless of the good or service considered. Although the logical framework is basically independent from the application adopted we strictly relate our description to CLASSIC [3, 4]. CLASSIC is a knowledge representation system that, although not endowed of a language expressive as more recent reasoners, *e.g.*, FaCT [17] and Racer [15], has polynomial-time inferences and, most important is a real system, is endowed of concrete datatypes and can be simply wrapped into a host system.

2 Description Logics basics

DLs [2, 10] are a family of logic formalisms whose basic syntax elements are *concept* names, *e.g.*,

person, degree, and *role* names, such as workingIn, requiredAS. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts. Formally, concepts are interpreted as subsets of a domain of interpretation Δ, and roles as binary relations (subsets of $\Delta \times \Delta$). Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DL include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations. Roles can be combined with concepts using *existential role quantification*, *e.g.*, Graduate $\sqcap$ $\exists$hasDegree.Engineering, which describes the set of graduates with an engineering degree, and *universal role quantification*, *e.g.*, person $\sqcap$ $\forall$livingIn.Apulia, which describes persons living exclusively in Apulia. Other constructs may involve counting, as number restrictions: Person $\sqcap$ ($\leq$ 1 Degree) expresses persons with at most one degree, and Person $\sqcap$ ($\geq$ 3 Specialization) describes persons endowed of at least three specializations. Many other constructs can be defined, increasing the expressive power of the DL, up to n-ary relations [6]. Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. For example we could impose that faculties can be divided into scientific and art ones using the two inclusions Faculty $\sqsubseteq$ Scientific $\sqcup$ SocialSciences and Scientific $\sqsubseteq$ $\neg$SocialSciences. Or that graduates has at least one degree as Graduate $\sqsubseteq$ ($\geq$ 1 hasDegree). Historically, sets of such inclusions are called TBox (Terminological Box). The basic reasoning problems for concepts in a DL are satisfiability, which accounts for the internal coherency of the description of a concept (no contradictory properties are present), and subsumption, which accounts for the more general/more specific relation among concepts, that forms the basis of a taxonomy. More formally, a concept C is satisfiable if there exists an interpretation in which C is mapped into a nonempty set unsatisfiable otherwise. If a TBox T is present, satisfiability is relative to the models of T, that is, the interpretation assigning C to a nonempty set must be a model of the inclusions in T. For instance, the concept AFaculty $\sqsubseteq$ Scientific $\sqcap$ SocialSciences is clearly unsatisfiable w.r.t. the TBox containing the inclusion scientific and social sciences. A concept C subsumes a concept D if every interpretation assigns to C a subset of the set assigned to D. Also subsumption is usually established relative to a TBox, a relation that we denote $T \models CD$. Also a TBox can be said satisfiable if there exist at least one model (i.e., an interpretation fulfilling all its inclusions in a nontrivial way). It is important to note that in the CLASSIC system we use, each C concept has an equivalent normal form as $C_{names} \sqcap C_{\sharp} \sqcap C_{all}$, in which C_{names} is a conjunction of names, $C_{\sharp}$ of number restrictions, and C_{all} of universal role quantifications. In the normal form, also all inclusions, definitions and disjoint groups have been made explicit [4]. CLASSIC provides the two basic reasoning services of DL-based systems, namely *Concept Satisfiability* (given a TBox T and a concept C, does there exist at least one model of T assigning a non-empty extension to C?), and *Subsumption* (given a TBox T and two concepts C and D, is C more general than D in any model of T?). Being a complete KR system, CLASSIC provides also data types as numbers and strings, and other services which are useful in a deployed prototype.

3 Principles for matchmaking

We highlight here common sense principles that a semantic based approach should yield. First of all, a matchmaking facilitator of practical use has to be liberal enough about details, without pretending a proposer to fill in forms with (say) 30 or more different characteristics to be set. This implies that the absence of a characteristic in the description of a requested or offered profile should not be interpreted as a constraint of absence. Instead, it should be considered as a characteristic that could be either refined later, or left open if it is irrelevant for a user — what is called *open-world assumption* in KR. Obviously, also the algorithm employed for matchmaking should take this issue into account. Secondly, a matchmaking system may give different evaluations depending on whether it is trying to match a request S with an offer D, or D with S — *i.e.*, depending on *who* is going to use this evaluation. This requirement is already evident when characteristics are modelled as sets of words. Of course, using sets of words to model supplies and demands would be too sensible to the choice of words employed — it misses meanings that relate words. It is now a common opinion that such fixed-terminology problems are overcome if terms have a logical meaning through an ontology [11]. Hence, we assume that supplies and demands are expressed in a DL. Obviously this approach includes the sets-of-keywords one, since a set of keywords can be considered also as a conjunction of concept names. We assume also that the common ontology is established, as a TBox in DL. Now a match between a supply S and a demand D could be evaluated according to T. Let $T \models \ldots$ denote logical implication (truth in all models of T), and let $\sqsubseteq$ (subsumption) denote also implication between constraints of S and D. There are three relations between concepts that we consider meaningful in semantic matchmaking: **Implication.** If $T \models (D \sqsubseteq S)$, then every constraint imposed by D is fulfilled (implied) by S, and vice versa if $T \models (S \sqsubseteq D)$. This relation extends the previous set-based inclusion to general concepts. If both $T \models (D \sqsubseteq S)$ and $T \models (S \sqsubseteq D)$, then D and S should be considered equivalent in T. This relation extends exact matching by ruling out irrelevant syntactic differences. **Consistency.** If $D \sqcap S$ is satisfiable in T, then there is a *potential* match, in the sense that the constraints of neither proposal exclude the other. This relation has been highlighted also by other researchers [26]. However, that proposal lacks a *ranking* between different potential matches, which we believe is fundamental in order to support *e.g.*, a project manager in the choice of the most interesting curricula, among all potential ones. **Inconsistency.** Otherwise, if $D \sqcap S$ is unsatisfiable in T, some constraints of one proposal are in contrast with the properties of the other one. However, also (say) supplies which are inconsistent with D may be reconsidered, if the demander accepts to *revise* some of D's constraints. We call this situation a *near miss* or *partial match*. The point of course is in revising not too much. Hence, also in this case a ranking — different from the one of potential matches — is fundamental.

We now state some properties that — we believe — every ranking function should have in logical matchmaking. First of all, a ranking for semantic matchmaking should be *syntax independent*. That is, for every pair of supplies S_1 and S_2, demand D, and ontology T, when S_1 is logically equivalent to S_2 then S_1 and S_2 should have the same ranking for D — and the same should hold also for every pair of logically equivalent demands D_1, D_2 with respect to every supply S. Secondly, a ranking for semantic matchmaking should be *monotonic over subsumption*. That is, for every demand D, for every pair of supplies S_1 and S_2, and ontology T, if S_1 and S_2 are both potential matches for D, and $T \models (S_2 \sqsubseteq S_1)$, then S_2 should be ranked either the same, or better than S_1. The same should hold also for every pair of demands D_1, D_2 with respect to a supply S. Intuitively, this property could be read of as "A ranking of potential matches is monotonic over subsumption if the more specific, the better." When turning to partial matches, adding another characteristic to an unsatisfactory proposal may either worsen its ranking (when another characteristic is violated) or keep it the same (when the new characteristic is not in contrast). Note that this ranking should be kept different from the ranking for potential matches. Obviously, properties pointed out here are independent of the particular

DL employed, or even the particular *logic* chosen. For instance, the same properties could be stated if propositional logic was used or also logics, such as DAML, for which reasoning systems are still unavailable.

4 Finding best matches

Here we present algorithms for matchmaking, which satisfy previously stated properties and have been devised adapting the original CLASSIC structural algorithm for subsumption [4]. Only the algorithm for potential match is completely outlined; the partial match one is similar, and is only briefly described here for space reasons. Recall that a CLASSIC concept C can be put in normal form as $C_{names} \sqcap C_{\sharp} \sqcap C_{all}$. Without ambiguity, we use the three components also as sets of the conjoined concepts. Moreover, recall that the TBox in CLASSIC can be embedded into the concepts, hence we do not consider explicitly the TBox, although it is present.

```
Algorithm $rankPotential(C, D)$;
input CLASSIC concepts $C$, $D$, in normal form, such that
      $C \sqcap D$ is satisfiable
output rank $n \geq 0$ of $C$ w.r.t. $D$, where 0 means that
      $C \sqsubseteq D$ (best ranking)
begin algorithm
  let $n := 0$ in
    /* add to $n$ the number of concept names in $D$ */
    /* which are not among the concept names of $C$ */
    1. $n := n + |D_{names+} - C_{names+}|$;
    /* add to $n$ number restrictions of $D$ */
    /* which are not implied by those of $C$ */
    2. for each concept $(\geq\ x\ R) \in D_{\sharp}$
       such that there is no concept $(\geq\ y\ R) \in C_{\sharp}$ with $y \geq x$
            $n := n + 1$;
    3. for each concept $(\leq\ x\ R) \in D_{\sharp}$
       such that there is no concept $(\leq\ y\ R) \in C_{\sharp}$ with $y \leq x$
            $n := n + 1$;
    /* for each universal role quantification in $D$ */
    /* add the result of a recursive call */
    4. for each concept $\forall R.E \in D_{all}$
           ifthere does not exist $\forall R.F \in C_{all}$
             then $n := n + rankPotential(\top, E)$;
             else $n := n + rankPotential(F, E)$;
    return $n$;
end algorithm
```

Obviously, total match is a particular case of potential match, obtained when $rankPotential(C, D) = 0$. With reference to the complexity, which is extremely important for a practical use of the system, the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up, as demonstrated by Nebel in [22]. And anyway, a polynomial algorithm cannot be expected since subsumption in $\mathcal{AL}$ with an acyclic TBox $\mathcal{T}$ is co-NP-hard [5]. However, in the cited paper Nebel argues that the expansion is exponential in the depth of the hierarchy $\mathcal{T}$; if the depth of $\mathcal{T}$ is $O(\log|\mathcal{T}|)$, then the expansion is polynomial, and so is the above algorithm. Notice that the algorithm penalizes generic profile descriptions, which in simple subsumption matching would be unfairly advantaged.

The algorithm for ranking partial matches follows again the partition of CLASSIC concepts into names, number restrictions, and universal role quantifications. However, this time we are looking for inconsistencies. Hence, when a universal role quantification is missing in either concept, the recursive call is unnecessary. For both algorithms it can be proved they respect the properties highlighted in the previous section.

5 A prototype system

The matchmaking framework presented in the previous sections has been deployed in a prototype facilitator. The system embeds a NeoClassic engine (a C++ implementation of the original CLASSIC system, whose sources have been adapted for our purposes).

The general architecture of the system is shown in Figure 2. The system can accept requests by both a dedicated client and by generic user agents as DAML+OIL descriptions.

The client is a Java applet (an extremely light client, in view of application on devices such as PDAs) that sends, via SOAP (http://www.w3.org/TR/SOAP/), an advertisement *i.e.*, a description of the request to be matched, as a string in KRSS (Knowledge Representation System Specification) (http://www.bell-labs.com/user/pfps/papers/krss-spec.ps) syntax, Figure 3 shows the content for an example advertisement in KRSS.

The SOAP packet contains the string and the URI of the reference ontology. The communication module is a web service whose main purpose is the translation of KRSS descriptions in portable DAML+OIL ones. The module, upon receipt of the packet, transforms the string in a DAML+OIL formatted description, using Jena APIs (http://www.hpl.hp.com/semweb/), Figure 4 shows the corresponding packet.

The description is forwarded to the Matchmaking service, which is the principal module of the architecture. It receives the SOAP packet, extracts the code and the URI that references the ontology requested. The matchmaking engine preliminarily checks for satisfiability w.r.t. the referenced ontology. If the check succeeds it carries out the matchmaking process with all descriptions in the repository corresponding to the given ontology, as will be described in detail in the next subsection.

The system accepts two types of requests, advertisement and query. For the first one, the system will store the request. In this way satisfiable queries/demands that remain unmatched will be automatically reexamined when new supplies are provided, and notification will be provided

Example Demand: (and APARTMENT (all HASROOMS BEDROOM) (at-most 0 HASPETS))

```
POST /soap/servlet/rpcrouter HTTP/1.0 Host: localhost:8070
Content-Type: text/xml Content-Length: 597 SOAPAction: ""

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"> <SOAP-ENV:Body>
<ns1:translate xmlns:ns1="urn:demo1:Translator"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<text xsi:type="xsd:string">(and APARTMENT (all HASROOMS BEDROOM)
(at-most 0 HASPETS))</text> <uri
xsi:type="xsd:string">http://www.example.org/rent-toy-ontology#</uri>
<requestType xsi:type="xsd:string">demand</requestType>
</ns1:translate> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

Figure 3. An example KRSS SOAP packet

```
POST /soap/servlet/rpcrouter HTTP/1.0 Host: localhost:8070
Content-Type: text/xml Content-Length: 1979 SOAPAction: ""

<SOAP-ENV:Envelope
xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/1999/XMLSchema"> <SOAP-ENV:Body>
<ns1:matchMake xmlns:ns1="urn:demo1:Matchmaker"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<text xsi:type="xsd:string"> <rdf:RDF
xmlns:rdf='http://www.w3.org/1999/02/22-rdf-syntax-ns#'
xmlns:daml='http://www.daml.org/2001/03/daml+oil#'
xmlns:rdfs='http://www.w3.org/2000/01/rdf-schema#'> <daml:Datatype
rdf:about='http://www.example.org/rent-toy-ontology#HASROOMS'/>
<daml:Datatype
rdf:about='http://www.example.org/rent-toy-ontology#BEDROOM'/>
<daml:Datatype
rdt:about='http://www.example.org/rent-toy-ontology#HASPETS'/>
<rdfs:Class
rdf:about='http://www.example.org/rent-toy-ontology#APARTMENT'/>
<daml:Property
rdf:about='http://www.daml.org/2001/03/daml+oil#maxCardinality'/>
<daml:Property
rdf:about='http://www.daml.org/2001/03/daml+oil#onProperty'/>
<daml:Property
rdf:about='http://www.daml.org/2001/03/daml+oil#toClass'/>
<daml:Restriction> <daml:onProperty
rdf:resource='http://www.example.org/rent-toy-ontology#HASROOMS'/>
<daml:toClass
rdf:resource='http://www.example.org/rent-toy-ontology#BEDROOM'/>
</daml:Restriction> <daml:Restriction daml:maxCardinality='0'>
<daml:onProperty
rdf:resource='http://www.example.org/rent-toy-ontology#HASPETS'/>
</daml:Restriction> </rdf:RDF> </text>
<uri
xsi:type="xsd:string">http://www.example.org/rent-toy-ontology#</uri>
<requestType xsi:type="xsd:string">demand</requestType>
</ns1:matchMake> </SOAP-ENV:Body> </SOAP-ENV:Envelope>
```

Figure 4. The DAML translation of the previous KRSS description

for successful match. The same service is available for unmatched supplies. The query service does not permanently store demand/supply descriptions.

The matchmaker output is forwarded according to specifications included int the SOAP packet and can be formatted as an e-mail, a SMS (Small Message System) or a JSP (Java Servlet Page), if the request has been submitted by the client. If the request has been submitted by an agent the response will be sent to the communication service for translation from KRSS to DAML+OIL.

Potential and partial matches can also trigger further communication services. For partial matches, on the basis of the matchmaker response, a demand can be revised, *e.g.*, relaxing some constraint. For potential matches the system can ask the supplier of the advertisement whether some features although not advertised are anyway available.

It should be noticed that the architecture can simply be modified to host other reasoners, such as FaCT or Racer. The rationale of the choice of the CLASSIC system, apart from the obviously useful availability of concrete datatypes and the possibility to extend its functionalities through test functions, is that its polynomial time inference allows practically synchronous operations, even with large ontologies. Recently DAML-S [24] has been proposed as a standard for the automation of web services and processes. It is a set of markup constructs based on DAML+OIL and we plan to implement support for it in the version of the system, where we will introduce also a negotiation service, which is currently not supported.

5.1 The matching engine

Our matching engine, whose architecture is shown in figure 1, is based on Java servlets; it embeds the NeoClassic reasoner and communicates with the reasoner running as a background daemon. At this stage of the work, the system is not fully transactional, so requests have to be serialized by the engine.

The system receives the KRSS string describing the demand/supply and the URI referencing the proper ontology. The Reasoner checks the description for consistency; if it fails, based on the reasoner output, the system provides an error message stating the error occurred. Otherwise the proper matchmaking process takes place. The NeoClassic standard [4] subsumption algorithm has been adapted in accordance with the matchmaking algorithms presented in section 4. Each match can return a 0, which means exact match or a value > 0. Recall that returned values for partial matches and potential matches have logically different meaning and matching descriptions are sorted in different sets. The matching engine may return then up to three separate result sets.

The matchmaker can also use weights to increase the relevance of concepts. The weight is a positive integer that keeps into account the occurrence of a role or a concept w.r.t. all Demands and Supplies available for a reference ontology. For each concept or role present in an advertisement, in normal form, that is $A \in D_{names+}$ or a concept $(\geq \ x \ R) \in D_{\sharp}$ or $(\leq \ x \ R) \in D_{\sharp}$ the corresponding weight is increased after each matching process. Obviously the matchmaking algorithm is modified in that increments are no more unitary but correspond to the assigned weights.

It is noteworthy that the ranking also prevents advertisements from being submitted in an extremely generic way. Simple subsumption matching [26] without ranking, in fact, favors *unfair* generic advertisements, which will be present in practically any retrieved set. Instead in our system, though logically potentially matching, the generic supply5 is given an high rank, which penalizes it.

6 Conclusion

We have addressed the problem of matchmaking of web services from a knowledge representation perspective. We proposed match categorization in terms of exact match, potential match and partial match and rank of matches within categories. In accordance with the above properties we have devised algorithms for matchmaking. The approach has been implemented in a matchmaking facilitator. The core engine has been implemented adapting the NeoClassic reasoner, and we have shown that, although with a reduced expressiveness w.r.t. more recent reasoners, CLASSIC can be effectively used for this class of problems.

References

[1] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001. available at: http://www.w3.org/2001/sw/.

[2] A. Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671-682, 1995.

[3] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989.

[4] A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.

[5] D. Calvanese. Reasoning with inclusion axioms in description logics: Algorithms and complexity. In *Proceedings of the Twelfth European Conference on Artificial Intelligence (ECAI'96)*, pages 303–307. John Wiley & Sons, 1996.

[6] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[7] F. Casati and M. C. Shan. Dynamic and Adaptive Composition of E-Services. *Information Systems*, 26:143–163, 2001.

[8] E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web services description language. available at: http://www.w3.org/TR/wsdl, 2001.

[9] E. Di Sciascio, F. Donini, M. Mongiello, and G. Piscitelli. A Knowledge-Based System for Person-to-Person E-Commerce. In *Proc. Workshop on Applications of Description Logics (KI 2001)*, 2001.

[10] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in Description Logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[11] D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.

[12] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463. ACM, 1994.

[13] Y. Gil and S. Ramachandran. PHOSPHORUS: a Task based Agent Matchmaker. In *Proc. AGENTS '01*, pages 110–111. ACM, 2001.

[14] J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. Workshop on Applications of Description Logics (KI 2001)*, september 2001.

[15] V. Haarslev and R. Moller. RACER System Description. In *International Joint Conference on Automated Reasoning, IJCAR'2001*, pages 701–705. Springer-Verlag, Berlin, 2001.

[16] Y. Hoffner, C. Facciorusso, S. Field, and A. Schade. Distribution Issues in the Design and Implementation of a Virtual Market Place. *Computer Networks*, 32:717–730, 2000.

[17] I. Horrocks. The FaCT system. In *Automated Reasoning with Analytic Tableaux and Related Methods: International Conference Tableaux'98*, number 1397 in Lecture Notes in Artificial Intelligence, pages 307–312. Springer-Verlag, Berlin, May 1998.

[18] I. Horrocks and et al. DAML+OIL language specification. available at: http://www.daml.org/2001/03/daml+oil-index, 2001.

[19] N. Karacapilidis and P. Moraitis. Building an Agent-Mediated Electronic Commerce System with Decision Analysis Features. *Decision Support Systems*, 32:53–69, 2001.

[20] D. Kuokka and L. Harada. Integrating Information Via Matchmaking. *Journal of Intelligent Information Systems*, 6:261–279, 1996.

[21] O. Lassila and R. R. Swick. Resource description framework. available at: http://www.w3.org/TR/REC-rdf-syntaxl, 1999.

[22] B. Nebel. Terminological Reasoning is Inherently Intractable. *Artificial Intelligence*, 43:235–249, 1990.

[23] T. D. Noia, E. D. Sciascio, F. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In *ACM SAC-03*, pages 582–586, Melbourne, Florida, USA, 2003.

[24] M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *The Semantic Web - ISWC 2002*, number 2342 in Lecture Notes in Computer Science, pages 333–347. Springer-Verlag, 2002.

[25] K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneus Ssoftware Agents in Cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.

[26] D. Trastour, C. Bartolini, and C. Priest. Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *Proc. WWW '02*, pages 89–98. ACM, 2002.

[27] H. Wang, S. Liao, and L. Liao. Modeling Constraint-Based Negotiating Agents. *Decision Support Systems*, 33:201–217, 2002.