A System for Principled Matchmaking in an Electronic Marketplace

Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, and Marina Mongiello

ABSTRACT: More and more resources are becoming available on the Web, and there is a growing need for infrastructures that, based on advertised descriptions, can match demands with supplies in an electronic marketplace. Several matchmakers rely on simple keyword matching, but significant matching results can only be obtained by exploiting the semantics inherent in structured descriptions. To this end, a novel knowledge representation approach is proposed, based on description logics, that is superior to both keyword- and basic subsumption-based matchmaking. The properties a matchmaker should have are formalized, and a matchmaking facilitator, compliant with the desired properties, is presented. The system embeds a NeoClassic reasoner whose structural subsumption algorithm has been modified to allow match categorization into potential and partial, and ranking of matches within categories. Experiments show good correspondence between users and system rankings.

KEY WORDS AND PHRASES: Categorization and ranking, commonsense reasoning, decision support, description logics, e-marketplace, matchmaking.

Several models of market transactions in an electronic marketplace have been proposed. Schmid and Lindemann's classification is based on four successive phases: information (*i.e.,* participants in the market seek potential partners), agreement (*i.e.,* negotiation on the terms of the agreement and agreement on a contract), settlement (i.e., payments and logistics), and after-sales (i.e., customer support) [41].

Most researchers have concentrated on the second and third phases of market transactions. In particular, a number of solutions for negotiation and brokerage have been investigated and proposed. Nevertheless the initial phase is no less important, since in order to begin negotiating a business transaction, one must search and find potential counterparts, at least in the dynamic scenario that a marketplace should be.

Matchmaking is the process of searching the space of possible matches between demand and supplies. This process is quite different from simply finding, once given a demand, a matching supply (or vice versa). Instead, it includes finding all the supplies that can fulfill the demand to some extent and identifying the most promising ones.

The authors thank Peter Patel-Schneider for valuable support on the CLASSIC system. This work was supported by MURST project CLUSTER 22, by EU-POP project "Negotiation Agents for the Electronic Marketplace," by PON project "Tecnologie innovative per la valorizzazione e la fruizione dei Beni Culturali," and by Italian CNR projects LAICO and "Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio."

International Journal of Electronic Commerce / Summer 2004, Vol. 8, No. 4, pp. 9–37.
Copyright © 2004 M.E. Sharpe, Inc. All rights reserved.
1086-4415/2004 $9.50 + 0.00.

The widespread availability of resources and services enables, among other advantages, interaction with a number of potential counterparts. The bottleneck is that it is difficult to find matches, possibly the most promising ones, between parties. Matchmaking is more difficult, yet challenging, when the counterparts are peer entities, users or agents. It is clear that bringing electronic commerce to its full potential requires a peer-to-peer (P2P) approach—anybody must be able to trade and negotiate with anybody else [20]. The scenario 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. The infrastructure should receive and store advertisement descriptions by both demanders and suppliers, and, as dynamically new demands or supplies are submitted, find and return the most satisfying matches. The infrastructure has to treat suppliers and demanders in a uniform way, and base the matches on common, extensible ontologies for describing both supplies and demands [21].

The uniform treatment of supply and demand calls for technical solutions quite different from the database ones usually employed in the business-to-consumer (B2C) scenario. In P2P matchmaking, the choice of which is the data and which is the query depends solely on a point of view—perhaps on who is more willing to actively search for the other partner.

Knowledge representation (KR), and in particular description logics (DL), can deal with the uniform treatment of knowledge from suppliers and demanders by modeling both as generic concepts to be matched. In fact, the logical approach on which DL is based allows for an open-world assumption. Incomplete information is allowed (and can be filled after a selection of possible matches), and the absence of information can be distinguished from negative information, making it possible to discard offers/requests that do not have the necessary properties, and to ask for missing information in the potential matches.

Matchmaking infrastructures have been proposed in the literature (see below for a survey), but they lack a formal framework that can logically classify and rank matches. The importance of ranking cannot be underestimated. It is of extreme importance for the practical use of any matchmaking system. Having hundreds of potential counterparts without indicating which of them is the most promising does not provide much added value to a facilitator system. The key questions to be answered in a dynamic framework are how far a given demand (or supply) is from a potential counterpart, and what requirements will eventually fulfill it. The answers to such questions have to rely on publicly available algorithms so as to encourage trust and prevent doubts about the fairness of the proposals returned by the matchmaking facilitator.

If supplies and demands were simple names or strings, the only possible match would be identity, resulting in an all-or-nothing approach to matchmaking. Although effective for fixed technical domains, such an approach misses the fact that supplies and demands usually have a structure of some sort. Such a structure could be exploited in order to evaluate "interesting" *inexact* matches. Vector-based techniques taken by classical

information retrieval (IR) can also be used, thus reverting matchmaking to a similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker or in LARKS [33, 44]. Obviously, a lack of document structure in descriptions would make matching only probabilistic and strange situations might ensue. Consider, for example, the simple demand "apartment with two rooms in Soho, pets allowed, no smokers" and the supply "apartment with two rooms in Soho, no pets, smokers allowed." Although in obvious conflict, they would correspond to a perfect match.

Our setting allows categorizing and rank matches according to their logical relation. In particular, it distinguishes between:

- *Exact match:* All requests in demand are available in supply (or vice versa).
- *Potential match:* Some requests in demand are not specified in supply (and further action—inquire—can be taken).
- *Partial match:* Some requests in demand are in conflict with supply (and further action—retract—can be taken).

Although the logical setting is independent from the application adopted, the proposed infrastructure is strongly related to CLASSIC [8, 9]. CLASSIC has a language that is less expressive than more recent reasoners (e.g., FaCT and Racer), but it is a knowledge-representation system that can make polynomial-time inferences. Moreover, and most important, it is a real system, with a rich API set and concrete data types that can be wrapped into a host system. The infrastructure currently embeds a modified version of NeoClassic, a C++ CLASSIC implementation, to carry out the matchmaking process.

The discussion in this paper refers throughout to an apartment-rental electronic marketplace, which was chosen as a case study. A subset of the ontology used as a reference in the examples is reported in Figure 1.

Related Work

The area of matchmaking has only recently become the subject of extensive investigation. Research pertaining to it, however, can be dated back to work on vague query answering that addressed the need to overcome limitations of relational databases with the aid of weights attributed to several search variables [36].

The earliest matchmakers, based on the KQML, were proposed by Finin, Fritzson, McKay, and McEntire and by Kuokka and Harada [22, 33]. They introduced matchmaking as an approach whereby potential producers/consumers could, either directly or through agent mediation, provide descriptions of their products/needs that would later be unified by a matchmaker engine to identify potential matches. Nevertheless, the proposed solutions to this challenging issue reverted to either a rule-based approach using the knowledge interchange format (KIF) (the SHADE prototype) or a free-text comparison (the COINS prototype), which deals with descriptions the same way as free-text retrieval tools [23, 33]. Standard information-retrieval

```
(createRole LOCATION)
(createRole HASPETS)
(createRole HASSERVICES)
(createRole OCCUPANTS)
(createRole HASROOMS)
(createRole HASPLACES)
(createRole COST true)[*]
(createConcept ROOM TOP true)
(createConcept PERSON TOP true)
(createConcept NEWYORK TOP true)
(createConcept SERVICES ROOM true)
(createConcept KITCHEN SERVICES true)
(createConcept WORKER PERSON true)
(createConcept STUDENT PERSON true)
(createConcept BATHROOM SERVICES true)
(createConcept SMOKER PERSON smoke)
(createConcept NON-SMOKER PERSON smoke)
(createConcept BED (and (at-least 1 OCCUPANTS) (at-most 1 OCCUPANTS) (all OCCUPANTS
PERSON)) true)
(createConcept BEDROOM (and ROOM (all HASPLACES BED) (at-least 1 HASPLACES)) true)
(createConcept SINGLE-ROOM (and BEDROOM (at-least 1 HASPLACES) (at-most 1
HASPLACES)) false)
(createConcept DOUBLE-ROOM (and BEDROOM (at-least 2 HASPLACES) (at-most 2
HASPLACES)) false)
(createConcept MULTIPLE-ROOM (and BEDROOM (at-least 3 HASPLACES)) false)
(createConcept APARTMENT (and (at-least 1 OCCUPANTS) (all OCCUPANTS PERSON) (at-
least 1 HASROOMS) (all HASROOMS ROOM) (at-least 2 HASSERVICES) (all HASSERVICES
SERVICES)) true)
```

Figure 1. The Example Ontology in CLASSIC
([*]COST is a functional role i.e., an attribute)

techniques have been also used in the recently proposed GRAPPA matchmaking framework [46]. Approaches similar to the cited ones were deployed in SIMS, which used KQML and LOOM as description languages, and InfoSleuth, which adopted KIF and the deductive database language LDL++ [1, 28] LOOM is also at the basis of the subsumption matching addressed by Gil and Ramachandran [24].

More recently there has been a growing interest in matchmaking engines and techniques, with the emphasis either on e-marketplaces or generic Web services, in view of the promised transformation of the human-understandable Web to a semantic (i.e., machine-understandable) Web. Significant examples include the work of Sycara, Widoff, Klusch, and Lu and of Paolucci, Kawamura, Payne, and Sycara, who propose a language, LARKS, specifically designed for agent advertisement [38, 44]. The matching process is carried out through five progressive stages, going from classical IR analysis of text to semantic matching via Θ subsumption. The notion, inspired by Software Engineering, of *plug-in* matches is introduced to overcome the limitations of a matching approach based on exact matches. No ranking is presented except for what so-called relaxed matches, which revert to an IR free-text similarity

measure. Therefore, a basic service of a semantic approach, such as inconsistency checking, seems unavailable with this type of match. An extension to the approach by Paolucci et al. is proposed by Li and Horrocks, who introduced two new levels for service profile matching [34]. They use the JADE agent platform for semantic Web services discovery on a test ontology based on DAML-S. Note that they introduce the *intersection satisfiable* level, whose definition is close to the one proposed here for potential matching. The approach presented does not introduce a ranking method to measure proximity of service descriptions.

Gonzales-Castillo, Trastour, and Bartolini and Priest propose a matchmaking framework that operates on service descriptions in DAML+OIL and is based on the FaCT reasoner [25, 45]. Unfortunately, as the authors admit, FaCT is very expressive but lacks concrete data types, which are obviously extremely useful for e-commerce applications, and their prototype is incomplete.

Semantic service discovery via matchmaking in the Bluetooth framework is investigated by Avancha, Joshi, and Finin [2, 5]. Here too the issue of somehow ranking and proposing approximate matches in the absence of exact matches is discussed, but as in the previous papers no formal framework is given. Instead, a logical formulation is expected to allow devising correct algorithms to classify and rank matches.

In addition, various current commercial electronic marketplaces try to provide matchmaking capabilities between demand and supply. Jango provides a system that allows comparison, but only in terms of price, of goods available in on-line stores on the Internet [19]. The description of the product to be matched has to be complete and consistent, and no reasoning on set containment or inconsistency check can be carried out. PersonaLogic allows customers to impose constraints for alternatives seeking [39]. It must be pointed out that constraints cannot be dynamically placed but have to be taken from a predetermined category set. Kasbah is a more effective system that allows dynamically setting constraints, but it does not allow handling of inconsistency and partial or potential matches [31]. A similar approach is also deployed in Tete-a-Tete [35]. A more advanced constraint-based approach, proposed by Karacapilidis and Moraitis, is able to handle conflicting preferences in demand/supply [30]. Consistency checking of preferences is accomplished by visiting an offer synthesis graph with a path consistency algorithm each time a new offer is entered. A further example is Smartclient, a system that allows users basic criteria adjustment, by presenting an interface that shows the initial search space, which can be reduced by further user interaction with the results [40]. The underlying system relies on partial constraint-satisfaction techniques. In a recent proposal along the same lines by Wang, Liao, and Liao, negotiation agents are formally modeled using an object-oriented constraint language [47].

IBM's Websphere matchmaking environment is, to our knowledge, the first example of a commercial solution that places explicit emphasis on the matchmaking between a demand and a supply in a peer-to-peer way, which is referred to as *symmetric* matchmaking [26]. As will be pointed out, the notion of symmetric matchmaking is questionable. The environment is based on a matchmaking engine that describes supplies/demands as properties and rules.

Properties are name-value pairs constructed using an extension of the Corba Trading service language. Rules are constructed using a generic script language. Matching is then accomplished by simply comparing properties and verifying rules. There is no notion of a distinction between full, partial potential and inconsistent matches.

Casati and Shan take a similar approach, with descriptions defined in XML and again a rule-based decision system [11]. Here descriptions of supply/demand can be stored in the e-service platform when a match is not available for further processing should a counterpart become available. Ströbel and Stolze propose an extension to the original Websphere matchmaker that introduces user specifications of negotiable constraints when no total match is available [43]. Thus their approach aims at some of the same issues addressed in this paper, but with a different, constraint-based, perspective. Di Sciascio, Donini, Mongiello, and Piscitelli present an initial setting for logical matchmaking in a person-to-person framework [16].

Matching in description logics has been extensively treated by Baader, Kusters, Borgida, and McGuinness, but not in relation to matchmaking [4]. Their work, in fact, considers expressions denoting concepts with variables in expressions. Then a match is a substitution of variables with expressions that makes a concept expression equivalent to another. In addition, the more general setting of concept rewriting in DLs has no direct relation to matchmaking.

Description Logics and the CLASSIC System

Description logics is a family of logic formalisms for knowledge representation [3, 7, 18]. Basic syntax elements are *concept* names (e.g., book, person, product, apartment), *role* names (e.g., author, supplier, hasRooms), and *individuals* (e.g., NewYorkCity, BackYardGarden, TVset 123. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts, such as the role author that links books to persons (their writers). Individuals are used for special named elements belonging to concepts.

More formally, a semantic *interpretation* is a pair $\Im = (\Delta, \cdot^{\Im})$, which consists of the *domain* Δ and the *interpretation function* $\cdot^{\Im}$, which maps every concept to a subset of Δ, every role to a subset of $\Delta \times \Delta$, and every individual to an element of Δ. We assume that different individuals are mapped to different elements of Δ, *i.e.*, $a^{\Im} \neq b^{\Im}$ for individuals $a \neq b$. This restriction is usually called *unique* name assumption (UNA).

Basic elements can be combined using constructors to form concept and role expressions, and each DL has its own distinct set of constructors. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$. Some DLs also include disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations. Roles can be combined with concepts using *existential role quantification (e.g.,* book $\sqcap \exists$author.italian, which describes the set of books whose authors include an Italian) and *universal role quantification* (e.g., product $\sqcap \forall$ supplier.japanese, which describes products sold only by Japanese suppliers). Other constructs may involve counting as number restrictions: apartment $\sqcap$ ($\leq$ 1 hasRooms) expresses apartments with just one room, and book

$\sqcap$ ($\geq$ 3 author) describes books written by at least three people. Many other constructs can be defined, increasing the expressive power of the DL, up to *n*-ary relations [10]. Expressions are given a semantics by defining the interpretation function over each construct. For example, concept conjunction is interpreted as set intersection: $(C \sqcap D)^{\Im} = C^{\Im} \cap D^{\Im}$, and the other boolean connectives $\sqcup$ and $\neg$, when present, are also given the usual set-theoretic interpretation of $\cup$ and complement. The interpretation of constructs involving quantification on roles needs to make domain elements explicit. For example, $(\forall R.C)^{\Im} = \{d_1 \in \Delta \mid \forall d_2 \in \Delta : (d_1, d_2) \in R^{\Im} \rightarrow d_2 \in C^{\Im}\}$.

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, one could impose that books can be divided into paperbacks and hardcover, using the two inclusions book $\sqsubseteq$ paperbacks$\sqcup$hardcover and paperbacks $\sqsubseteq \neg$ hardcover. Or, that books have only one title as book $\sqsubseteq$ ($\leq$ 1 title). Definitions are useful to give a meaningful name to particular combinations, as in doubleRoom $\equiv$ room$\sqcap$ ($=$ 2 hasPlaces). Historically, sets of such inclusions are called TBoxes (terminological boxes). In simple DLs, only a concept name can appear on the left-hand side of an inclusion.

The semantics of inclusions and definitions is based on set containment. An interpretation $\Im$ satisfies the inclusion $C \sqsubseteq D$ if $C^{\Im} \subseteq D^{\Im}$, and it satisfies the definition $C = D$ when $C^{\Im} = D^{\Im}$. A *model* of a TBox, T, is an interpretation satisfying all inclusions and definitions of T.

In every DL-based system, at least two basic reasoning services are provided:

- *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?
- *Subsumption:* Given a TBox T and two concepts C and D, is C more general than D in any model of T?

The CLASSIC system was developed at AT&T Bell Laboratories, where it has been adopted in several projects about configuration and program repositories [8, 9, 14, 48]. Its language was designed to be as expressive as possible while still admitting polynomial-time inferences. Therefore, it provides intersection of concepts but no union, universal but not existential quantification over roles, and number restrictions over roles but no intersection of roles, since each of these combinations is known to make reasoning NP-hard [17]. The discussion in this paper only considers the constructs that can be given a declarative semantics, and ignores those that allow one to make use of the host programming language.

The CLASSIC language has several constructs that are peculiar for description logics. First, there is the construct ONE-OF($a_1, \ldots, a_k$), which intuitively stands for the set of individuals $a_1, \ldots, a_k$. Second, CLASSIC distinguishes two kinds of roles: usual roles (denoted as P) and functional roles, called *attributes* (denoted as F). Both kinds of roles can be employed in expressions of the form FILLS(P, a) and FILLS(F, a), which intuitively describe the set of objects having the individual a as a filler of the role P or the attribute F, respectively. Finally, attributes can be combined into chains $F_1 \circ \ldots \circ F_n$ (denoted as p, q). Such chains

Name	System notation	Syntax	Semantics
top	top	$\top$	$\Delta^{\mathfrak{I}}$
bottom		$\bot$	$\varnothing$
intersection	(and C D)	$C \sqcap D$	$C^{\mathfrak{I}} \cap D^{\mathfrak{I}}$
universal quantification	(all R C)	$\forall R.C$	$\{ d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathfrak{I}} \rightarrow d_2 \in C^{\mathfrak{I}} \}$

Table 1. Syntax and Semantics of the Constructs of CLASSIC.

Name	System notation	Syntax	Semantics
definition	(createConcept A C false)	$A = C$	$A^{\mathfrak{I}} = C^{\mathfrak{I}}$
inclusion	(createConcept A C true)	$A \sqsubseteq C$	$A^{\mathfrak{I}} \subseteq C^{\mathfrak{I}}$
disjoint group	(createConcept A_1 C *symbol*).. (createConcept A_k C *symbol*)	$\text{disj}(A_1,\ldots,A_k)$	$A_i^{\mathfrak{I}} \cap A_j^{\mathfrak{I}} = \varnothing$ for $i \neq j$

Table 2. Syntax and Semantics of the TBox CLASSIC Assertions.

Note: symbol is a name denoting the group of disjoint concepts.

can appear in concepts of the form SAME-AS(p, q), which are interpreted as the set of objects such that the chain p leads to the same object as the chain q.[1]

The declarative core of CLASSIC's concept language is summarized in Table 1, which shows the syntax, both the abstract one and the actual CLASSIC one, and the semantics of constructs.

The semantics is as one would expect, except for the constructs ONE-OF and FILLS, that make it possible to refer to individuals. Individuals appearing in these expressions have a semantics different from individuals in first-order logic. They are interpreted as primitive disjoint concepts [9], i.e., as subsets of the domain, instead of as single elements of it. According to the authors, the rationale for this non-standard choice is that, otherwise, the properties of individuals would influence subsumption. Every CLASSIC concept can be given a *normal form*. The normal form of an unsatisfiable concept is simply $\bot$. Here the normal form is presented only for the constructs used in the ontologies and applications, namely, conjunction, number restrictions, and universal role quantifications. Every satisfiable concept C can be divided into three components: $C_{\text{names}} \sqcap C_{\#} \sqcap C_{\text{all}}$. The component C_{names} is the conjunction of all concept names $A_1 \sqcap \ldots \sqcap A_h$. The component $C_{\#}$ is the conjunction of all number restrictions, no more than two for every role (the maximum at-least and the minimum at-most for each role), including for every conjunct of C of the form $\forall R.\bot$, the number restriction $(\leq 0\ R)$ in $C_{\#}$. The component C_{all} conjoins all concepts of the form $\forall R.D$, one for each role R, where D is again in normal form. The fact that there is just one universal role quantification for each role is justified by the equivalence $\forall R.D_1 \sqcap \forall R.D_2 \equiv \forall R.D_1 \sqcap D_2$.

Moreover, the TBox in CLASSIC can be embedded into the concepts by expanding definitions and adding the right-hand-side concepts of inclusions, and adding the negation of disjoint concept names as shown in Table 2. For instance, suppose that a TBox contains:

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 2. A Sample KRSS SOAP Packet

1. the definition $\mathsf{doubleRoom} \doteq \mathsf{room} \sqcap (= 2\ \mathsf{hasPlaces})$
2. the inclusion $\mathsf{apartment} \sqsubseteq (\geq 1\ \mathsf{hasRooms})$
3. the disjointness assertion disj(apartment,room)

Then, the concept $\mathsf{apartment} \sqcap \forall \mathsf{hasRooms}.\mathsf{doubleRoom}$ can be rewritten as $\mathsf{apartment} \sqcap \neg\mathsf{room} \sqcap (\geq 1\ \mathsf{hasRooms}) \sqcap \mathsf{hasRooms}.(\mathsf{room} \sqcap \neg\mathsf{apartment} \sqcap (= 2\ \mathsf{hasPlaces}))$, which is equivalent to the former with respect to models of the TBox. Observe that the concept name apartment is kept in the rewriting, since the inclusion gives only a necessary condition (≥ 1 hasRooms). The latter concept can be safely conjoined to apartment—making the inclusion unnecessary—but cannot replace it, since (≥ 1 hasRooms) is not a sufficient condition for apartment. Instead, $\mathsf{room} \sqcap (= 2\ \mathsf{hasPlaces})$ replaces doubleRoom (after conjoining $\neg$apartment for the disjointness assertion), since it is a necessary and sufficient condition for it. Once this rewriting has been carried over all concepts, the TBox can be safely ignored. As proved by Nebel, there are cases in which this rewriting leads to an exponential blow-up [37]. However, typical TBoxes are much different from these cases, and for them the rewriting has size polynomial in the TBox.

The normal form of concepts can take the TBox embedding into account. In this case, the component C_{names} of a concept C contains concept names C_{names^+} and negations of concept names $C_{\text{names}\ \neg}$.

In the current setting, only a subset of CLASSIC constructors is used, that is, those needed in an *ALN* (attributive language with unqualified number restrictions) logic. Also, note that we revert to CLASSIC host language test-functions calls when dealing with functional roles, such as cost.

Knowledge-Representation Approach to Matchmaking

Let us start with an example. Consider the following: demand D = {apartment,soho,twoRooms} and a generic supply S = {apartment,soho,boiler, twoRooms,quiet,noPets}. Although S and D are not identical, the fact that D is included in S tells the demander that S fulfills every constraint imposed

by D. Hence, from the point of view of the demander, *S* completely satisfies *D*. However, note that the explicitly stated constraints in *D* do not completely satisfy *S*, although they do not exclude *S* either. It could be the case that asking the demander to refine *D* would achieve an exact match. This example already highlights some of the properties that are meaningful for a matchmaking facilitator.

> *DEFINITION 1. [Closed- and Open-world] A matchmaking system uses closed-world descriptions if the absence of a characteristic in the description of a supply or a demand is interpreted as a constraint of absence. Otherwise, the system uses open-world descriptions.*

We opt for open-world descriptions, because in this case incomplete specifications are allowed, with details that could be either refined later or left open if they are irrelevant for a user. Of course, treating open-world descriptions requires, on one side, that the interface for describing requests to the matchmaking facilitator should be liberal enough about details, and on the other side, that the algorithm employed for matchmaking should consider this issue.

> *DEFINITION 2.* [Symmetric and non-symmetric evaluation] A matchmaking system uses symmetric evaluations when it considers in the same way the match of a supply S with a demand D, and D with S. Otherwise, it uses non-symmetric evaluations.

In non-symmetric matchmaking, the system may give different results depending on who is going to use the evaluation. This latter property is preferred to symmetry, because non-symmetry seems to yield a better service to the user looking for counteroffers. This was evident in the above example, where all the constraints of *D* were fulfilled by *S* but not vice versa. Hence, *S* should be among the top-ranked supplies in the list of potential partners of the demander, while *D* should not appear at the top in the list of potential partners of the supplier.

Of course, the sets-of-words approach would be too sensitive to the choice of words employed to be successfully used. It misses meanings that relate words. In the apartments rental scenario, a matching facilitator should take into account that "boiler" is a form of heating system, or that a constraint "no-pets" applies also to a dog. It is now commonly accepted that such fixed-terminology problems are overcome if terms have a logical meaning through an ontology [21]. Hence, from now on suppose that supplies and demands are expressed in a description logic DL, equipped with a model-theoretic semantics. Note that this approach includes the sets-of-keywords one, since a set of keywords can also be considered as a conjunction of concept names. For example, the set {apartment,soho,twoRooms} can be equivalently considered as $\mathsf{apartment} \sqcap \mathsf{soho} \sqcap \mathsf{twoRooms}$ without modeling the structure of concepts. Entering into the structure of concepts may yield to $\mathsf{apartment} \sqcap \forall \mathsf{location}.\mathsf{soho} \sqcap (= 2\ \mathsf{hasRooms})$, while a keyword "noPets" can be given a logical meaning as $\forall \mathsf{occupants}.(\leq 0\ \mathsf{hasPets})$.

Suppose, as well, that a common ontology for supplies and demands is established as a TBox in DL. Now a match between a supply S and a demand D can be evaluated according to T. Let $T \models \ldots$ denote logical implication (truth in all models of T), and let $\sqsubseteq$ (subsumption) denote implication between constraints of S and D. There are three relations between concepts expressing supplies and demands, that can be considered meaningful in 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 also been highlighted by other researchers [34, 45]. However, those proposals lack a *ranking* between different potential matches, which is fundamental in order to support a user in the choice of the most promising of all the potential partners.

Inconsistency

Otherwise, if $D \sqcap S$ is unsatisfiable in T, some constraints of one proposal are in conflict with the properties of the other one. However, when (say) a demand D that has no potential matches also supplies S that are inconsistent with D, it may be reconsidered, if the demander agrees to revise some of D's constraints. The point, of course, is in revising not too much. Hence, in this case too, a ranking different from the one of potential matches is fundamental, in order to highlight the least unsatisfactory proposals, which we call *near misses* or *partial matches.*

The discussion will now highlight some properties that every ranking function should have in logical matchmaking. These properties are stated in the form of definitions, so as to distinguish rankings having the property from rankings that do not.

First, if a logic is used to give some meaning to descriptions of supplies and demands, then proposals with the same meaning should have the same ranking, independently of their syntactic descriptions.

> *DEFINITION 3. Syntax independence in ranking potential matches.* A ranking of potential matches is syntax-independent if 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 have the same ranking for D, and the same holds for every pair of logically equivalent demands D_1, D_2 with respect to every supply S.

For example, an apartment S_1, described as available for the summer quarter, should have the same rank with respect to a request as another S_2 that is identical except that it is described as available for June–July–August. A similar property should also hold for ranking incoherent pairs of supplies and demands. The rationale of this property is that, in looking for least unsatisfactory proposals when recovering from an initial "no potential match," a partial match should be as "not-so-bad" as any equivalent one.

> *DEFINITION 4. Syntax independence in ranking partial matches.* A ranking of partial matches is syntax-independent if 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 have the same ranking for D, and the same holds for every pair of logically equivalent demands D_1, D_2 with respect to every supply S.

Even if this definition is very similar to the preceding one, they are kept separate because we envisage *two* rankings, one evaluating potential matches, and the other partial ones. In principle, one could think about matchmaking systems that are syntax-independent for potential matches but not for partial ones, or vice versa. Here syntax independence is chosen for both partial and potential matches. Clearly, when the logic admits a normal form of expressions—as CNF or DNF for propositional logic, or the normal form of concepts for the DL of CLASSIC mentioned earlier—using such a normal form ensures by itself syntax independence. We now consider the relation between ranking and implications. Let us go back to the descriptions with sets of words, since they are easy to read through. Let D be a demand and S_1, S_2 be two supplies defined as follows:

$$D = \{\text{apartment,soho,twoRooms,petsAllowed}\}$$

$$S_1 = \{\text{apartment,soho,boiler,quiet}\}$$

$$S_2 = \{\text{apartment,soho,boiler,quiet,lastFloor}\}$$

In this case, the characteristics that S_2 adds to S_1 are irrelevant for D. Hence, whatever the rank for S_1, the one for S_2 should be the same. If instead we let

$$S_3 = \{\text{apartment,soho,boiler,quiet,petsAllowed}\},$$

then S_3 should be ranked better than S_1, since it adds a characteristic required by D. This example from sets can be generalized, and the following definition can be stated:

> *DEFINITION 5. Monotonicity of ranking potential matches over subsumption.* A ranking of potential matches is monotonic over subsumption whenever 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 as or better than S_1,

and the same should hold for every pair of demands D_1, D_2 with respect to a supply S.

Intuitively, the above definition could be read: *A ranking of potential matches is monotonic over subsumption if the more specific the better.*

Observe that the word "better" is used instead of the symbols $\leq$ and $\geq$. This is because some rankings may assume that "'better" means "increasing" (toward infinity or unity), whereas others may assume "decreasing" (toward zero). When turning to partial matches, in which some properties are already in conflict between supply and demand, the picture reverses. Adding another characteristic to an unsatisfactory proposal may only worsen the ranking (when another characteristic is violated) or keep it the same (when the new characteristic is not in conflict). Note that this ranking should be kept different from the ranking for potential matches. After all, agreeing to discard one or more of the characteristics that we required is much worse than deciding among a ranked list of potential matches.

DEFINITION 6. Antimonotonicity of ranking partial matches over implication. A ranking of partial matches is antimonotonic over implication whenever, 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 partial matches for D, and $T \models (S_2 \sqsubseteq S_1)$, then S_2 should be ranked either the same as or worse than S_1, and the same should hold for every pair of demands D_1, D_2 with respect to a supply S.

Intuitively, the above property could read: *A ranking of partial matches is antimonotonic over subsumption if the more specific, the worse.* We stress here that the two rankings for potential and partial matches are different, and that partial matches are considered separately from potential ones. For partial matches, only if a user is willing to revise the demand (supply), in order to move from a partial match, with something in contrast, to a potential match with a supply (demand), the ranking for potential matches is applied to the revised offer.

The properties and definitions stated in this section 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 to describe supplies, demands, and the ontology. In this respect, this section retains its significance if one chooses expressive DLs like *SHOQ*(*D*) [26] or even logics for which representation and reasoning system are not yet fully available, like DAML [38].

Matchmaking Infrastructure

Using the highlighted properties as a formal specification, a prototype facilitator was designed and implemented. The system embeds a modified NeoClassic reasoner. NeoClassic is a C++ implementation of the original CLASSIC. The changes affected the structural subsumption algorithm, which was modified to compute matches classification and ranking, as will be shown later on.

The general architecture of the facilitator is pictured in Figure 3. Its main components are the MatchMaker Service (MMS) and the Communication Service (CS). The system can accept requests by a heavyweight client, a lightweight client, and a generic user agent.

The heavyweight client is a Java applet that allows advertisement description in natural language. In its current implementation, it is not a general-purpose application, in that its structure depends on the reference ontology because of the inherent contextualization. Currently the only supported ontology is the apartment-rental one used in the case study. The input for the Natural Language Interpreter (NLI) module is a free-text description of an advertisement for apartment rental. The output is a string formatted in KRSS (Knowledge Representation System Specification) of the input sentence [32]. To perform a better natural-language semantic analysis using context information, the grammar rules contain the SEM feature, whose values refer to a set of fundamental categories identified in the reference ontology (e.g., AP for *apartment,* ROOM for *rooms*, ACC for *accessories*). In this way, the grammar simultaneously represents both syntactic and semantic behavior of the analyzed context. The computational approach is chart-based. The algorithm can recognize unknown terms and ask for synonyms. If the new terms introduce new information in the system (i.e., information not described in the ontology), this last is updated. The unique constraint is that new information has to be related to accessories in the current settings. Once the ontology has been updated, the system lets advertisers know that they can update their advertisements using the new available information. The lightweight client is still a Java applet, but an extremely light one, in view of its application on devices such as PDAs that send, via SOAP, an advertisement (i.e., a description of the request to be matched) as a string in KRSS syntax [42]. Figure 2 shows an example advertisement in KRSS.

The SOAP packet contains the string and the URI of the reference ontology. The CS module is a Web service (registered with XMethods) whose main purpose is the translation of KRSS descriptions into portable DAML+OIL ones. The module, upon receipt of the packet, transforms the string in a description formatted in DAML+OIL, using Jena APIs [29]. Figure 4 shows the corresponding packet.

The system can also receive advertisements through a generic agent. In this case, they are accepted only if expressed in DAML+OIL. The CS translates the demands and supplies from KRSS to DAML+OIL. The description is forwarded to the MatchMaker Service, which is the principal module of the architecture. It receives the SOAP packet, and extracts the code and the URI that references the requested ontology. As a preliminary, the matchmaking engine checks for satisfiability with respect to the referenced ontology. If the check succeeds, it carries out the matchmaking process with all the descriptions in the repository corresponding to the given ontology, as will be described in detail further on.

The system accepts two types of requests, advertisements and queries. For the first type, 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 for successful matches. The same service is available for unmatched supplies. The query service,

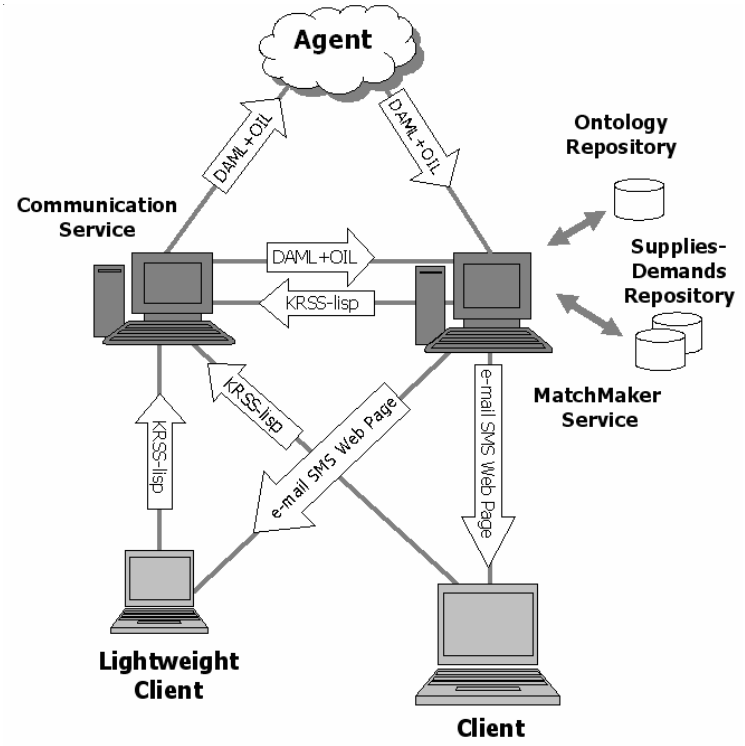

Figure 3. System Architecture of the Facilitator Infrastructure

instead, does not permanently store demand/supply descriptions. The matchmaker output is forwarded according to specifications included in the SOAP packet and can be formatted as an e-mail, a GSM SMS (Small Message System), or a JSP, if the request has been submitted by a 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.[2] For partial matches, based on the matchmaker response, a demand can be revised—for example, relaxing (retracting) a constraint. For potential matches, the system can ask the supplier of the advertisement whether certain features are available even though not advertised. The architecture could 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 data types and the possibility of extending its functionalities through test functions, is that its polynomial time inference allows practically synchronous operations even with large ontologies.

The Matching Engine

The matching engine is based on Java servlets. It embeds the NeoClassic reasoner and communicates with the reasoner running as a background dae-

```
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 rdf: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

mon. 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 returns an error message that is forwarded to the client/agent originating the request. Otherwise, the proper matchmaking process takes place. To this end, two algorithms have been devised, based on a modification of the original CLASSIC structural subsumption algorithm, to classify and determine a score for matches [9]. The rationale and the structure of the algorithms are described below.

A CLASSIC concept C can be put in normal form as $C_{\text{names}} \sqcap C_{\#} \sqcap C_{\text{all}}$. Without ambiguity, the three components can also be used as sets of the conjoined concepts. Moreover, recall that the TBox in CLASSIC can be embedded into the concepts. Hence, the TBox is not considered explicitly, although it is present. The algorithm easily follows a structural subsumption algorithm, except for the treatment of universal role quantification, which will now be explained with the help of an example. Suppose a demand D = apartment$\sqcap\forall$hasRooms.(singleRoom$\sqcap$nonSmokerRoom) and two supplies C_1, C_2, defined as follows:

C1 = apartment ⊓ ∀hasRooms.roomWithTV

C2 = apartment

Now, comparing D with C_1, a recursive call through the universal role quantification highlights that rooms in C_1 miss both characteristics required by D. Thus the ranking should be 2 (where ranking 0 would mean subsumption). In the case of C_2, instead, since the universal role quantification is absent, no recursive comparison is possible. However, observe that from the semantics, ∀hasRooms. ⊤ ≡ ⊤ (no restriction on the fillers of role hasRooms is equivalent to no restrictions at all). Thus, apartment ≡ (apartment ⊓ ∀hasRooms. ⊤). Since we want to enforce syntax independence, both concepts should yield the same ranking. Therefore we compare the last concept, which allows us to make a recursive comparison of characteristics of universal role quantifications.

Algorithm rankPotential (C, D);
input CLASSIC concepts C, D, in normal form,
such that C ⊓ D is satisfiable
output rank n ≥ 0 of C with respect to D, where 0 means
that C ⊑ 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 (≥ x R) ∈ $D_{\#}$
such that there is no concept (≥ y R) ∈ $C_{\#}$ with y ≥ x
n : = n + 1;
3. **for each** concept (≤ x R) ∈ $D_{\#}$
such that there is no concept (≤ y R) ∈ $C_{\#}$ with y ≤ x
n : = n + 1;
/* for each universal role quantification in D */
/* add the result of a recursive call */
4. **for each** concept ∀R.E ∈ D_{all}
if there does not exist ∀R.F ∈ C_{all}
then n : = n + rankPotential(⊤ , E);
else n : = n + rankPotential(F, E);
return n;
end algorithm

It is easy to modify the algorithm if the weights on subconcepts of D are taken into account. Instead of adding 1 to n for each of D's concepts missing in C, we just add the corresponding weight. In this way, when the proposal concerns apartments, the concept apartment gets the highest weight, and minor characteristics get lower weights. Then, a far rank would mean that either many minor characteristics or one very important characteristic has been left unspecified in C.

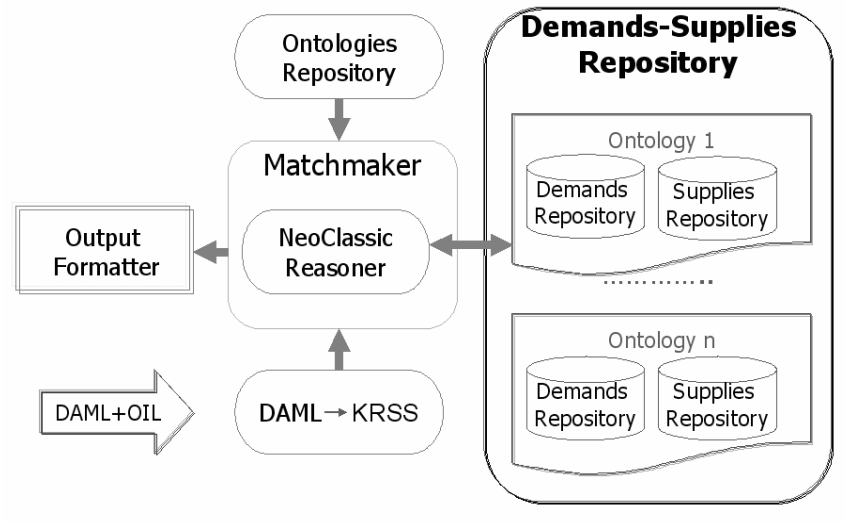

Figure 5. The Architecture of the Matchmaking Engine

A version of the algorithm was also implemented in which weights are *learned* by the system upon repeated analysis of proposals. In this case, of course, the learned weights are *absolute* ones, and not relative to a particular actor.

The algorithm for ranking partial matches follows the partition of CLASSIC concepts into names, number restrictions, and universal role quantifications. However, this time we are looking for inconsistencies. Hence, when universal role quantification is missing in either concept, the recursive call is unnecessary. In fact, suppose that $\forall R.E \in D_{\text{all}}$ while no quantification on R is present in C_{all}. No quantification is equivalent to the concept $\forall R.\top$, and a recursive call yields a comparison between $\top$ and E, which can only yield a 0 rank since $\top$ is consistent with every concept. Recall also that an inconsistency arising from $(\geq x\ R)$, with $x \geq 1$ in one concept, and $\forall R.\bot$ in the other concept, is already evidenced in the comparison of number restrictions, because $(\leq 0\ \text{R})$ has been added to the normal form of the latter concept.

Algorithm rankPartial (C, D);
input CLASSIC concepts C, D, both in normal form
output rank n ≥ 0 of C with respect to D,
where 0 means that C ⊓ D is satisfiable
begin algorithm
let n : = 0 **in**
/* add to n the number of concept names in C */
/* which are disjoint from the concept names of D */
for each concept name $A \in C_{\text{names+}}$
if there exists a concept $\neg A \in D_{\text{names?}}$

then n : = n + 1;
/* add to n number restrictions of C */
/* which are in conflict with those of D */
for each concept (≥ x R) ∈ $C_{\#}$
such that there is a concept (≤ y R) ∈ $D_{\#}$ with y < x
n : = n + 1;
for each concept (≤ x R) ∈ $C_{\#}$
such that there is a concept (≥ y R) ∈ $D_{\#}$ with y > x
n : = n + 1;
/* for universal role quantifications in C and D */
/* which are triggered by an at-least number restriction on
either concept */
/* add the result of a recursive call */
for each concept ∀ R.F ∈ C_{all}
if either (there exist both)
(≥ x R) ∈ $C_{\#}$ with x ≥ 1 **and** ∀R.E ∈ D_{all} with E≠ ⊥)
or (F ≠ ⊥ **and there exist both** (≥ x R) ∈ $D_{\#}$ with x ≥ 1
and ∀R.E ∈ D_{all})
then n : = n + rankPartial(F,E);
return n;
end algorithm

In this case too, weights could be added to subconcepts of *D*, where the greater the weight, the more that characteristic is important, making the rank of *C* far off when in conflict.

For both algorithms, weights can be determined according to their relevance/probability with various approaches, including, for example, approaches based on utility theory [13]. In this perspective, our framework can benefit, rather than serve as an alternative to, other methods for knowledge elicitation.

It can be proved that both algorithms respect the properties highlighted in the preceding section. With reference to complexity, as is well known, the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up [37]. Nevertheless, the expansion is exponential in the depth of the hierarchy of the TBox *T*. If the depth of *T* is $O(\log |T|)$, then the expansion is polynomial, and so too the algorithm [15].

Each match can return a 0, which means exact match or a value greater than zero. Recall that returned values for partial matches and potential matches have logically different meanings, and matching descriptions are sorted in different sets. The matching engine may then return up to three disjoint result sets, with results ranked in the potential and partial sets.

Both algorithms actually return score values, which can obviously be sorted to determine a ranking. Hence, the algorithms are consistent with the monotonicity (antimonotonicity) property already described. But they also assess a numeric score, which induces a total order, whereas qualitative criteria induce only a partial order on matches.

As pointed out above, the matchmaker can also use weights to increase the relevance of concepts. The weight is a positive integer that keeps account of the occurrences of a role or a concept with respect to all demands and supplies

available for a reference ontology. For each concept or role present in an advertisement, in normal form, that is $A \in D_{\text{names}^+}$ or a concept $(\geq x\ R) \in D_{\#}$ or $(\leq x\ R) \in D_{\#}$, the corresponding weight is increased after each matching process. The matchmaking algorithm is modified in that increments are no longer unitary but correspond to the assigned weights.

Note that users of the system do not have to be explicitly aware of the presence of weights. Therefore no further effort is asked of the user with respect to the basic version of the system. To simplify the reading, no weights are used in the following subsection.

Matchmaker Behavior

In order to show the behavior of the matchmaker consider, with reference to the toy ontology used throughout the paper, the example demand *apartment with bedroom wanted, no pets.* Also suppose the following advertisements have been previously submitted as supplies: *supply1: single room in an apartment of smoking persons; supply2: two double rooms in an apartment, no pets allowed; supply3: apartment to let, with a pet; supply4:lodging with bathroom, two pets*[3]*; supply5: apartment.* The translation of these advertisements in CLASSIC, in accordance with our ontology is pictured in Figure 6. The NeoClassic modified engine processes descriptions according to their explicit normal form, which are shown for the previous advertisements in Figure 7. From the application of the rankPartial algorithm, the rankings in Table 3 ensue.

This result points out that demand cannot be satisfied by supplies 3 and 4. The system provides a rank that indicates *how far* the demand is with reference to the supplies—that is, how much the demand should be revised. The system can also explicitly return details of concepts unsatisfied for the given demand. In this way the user can decide to revise the demand advertisement.

The remaining supplies are all *potential* matches. That is, there is nothing in them in conflict with demand. Yet it is obvious that a user needs to know which supplies may best match the demand. The execution of the *rankPotential* algorithm provides the answer.

The results in Table 4 show that supply2 is an exact match with respect to the given demand. Supplies 1 and 5 are potential matches, since there is no characteristic in conflict but some of the requests in demand are not explicitly available. When no exact match is available, the user may then carry out further investigations on potential matches. In this process, the user is again helped by the system ranking, which represents the distance between the demand and the supply.

The ranking also prevents advertisements from being submitted in an extremely generic way. Simple subsumption matching without ranking, in fact, favors *unfair* generic advertisements, which will be present in practically any retrieved set [45]. Instead, in the proposed system, though logically potentially matching, the generic supply5 is given a high rank, which penalizes it. The nonsymmetric behavior of the matchmaking process can also be highlighted here by simply exchanging the arguments of the matchmaking algorithms. By taking, for example, the description of supply1

demand : (and APARTMENT (all HASROOMS BEDROOM) (at-most 0 HASPETS))
supply1: (and APARTMENT (all HASROOMS SINGLE-ROOM) (all OCCUPANTS SMOKER))
supply2: (and APARTMENT (at-least 2 HASROOMS) (at-most 2 HASROOMS) (all HASROOMS DOUBLE-ROOM) (at-most 0 HASPETS))
supply3: (and APARTMENT (at-least 1 HASPETS))
supply4: (and (at-most 1 HASSERVICES) (all HASSERVICES BATHROOM) (at-least 2 HASPETS))
supply5: (and APARTMENT)

Figure 6. Demand and Supplies to Be Matched

and matching it against the example demand, the *rankPotential* algorithm returns a rank 4.

Experiments

A small experiment was set up to evaluate the degree of correspondence between the matchmaking facilitator rankings and users' perceptions.

All the apartment-rental advertisements were selected from the dedicated section of a local newspaper on October 6, 2002. They were subdivided in two sets, *demands* (23 advertisements) and *supplies* (39 advertisements). Twenty volunteers of both sexes and various ages were given a questionnaire that included 12 items. Each item was a demand and a set of up to eight supplies, or a supply and a set of up to eight demands. The volunteers were asked to rank, according to their judgment, elements of each set with respect to the given advertisement, with the following question: "Order the following demands (supply) with respect to the given supply (demands) as you would contact them had you issued the given supply (demand)." The volunteers were given unlimited time, and on average it took approximately a half-hour to complete the questionnaire.

The authors chose groups of advertisements such that the items included only potential matches, only partial matches, and both potential and partial matches. At least for the single-day pick, there were no exact matches. The same sets of items were submitted to the matchmaking engine. While the volunteers gave strictly ranked orderings, the system could also provide equal ranking for various matches. Anyway, as averaged user rankings were also considered, classification could provide equal ranking for various matches as a result of the averaging process.

The resulting matchmaker classification gave a "system-provided ranking." The R_{norm} was adopted as quality measure of the correspondence between system-provided and user-provided rankings [6]. R_{norm} can be defined as follows: Let G be a finite set of items (a supply and a set of demands or a demand and a set of supplies) with a user-defined preference relation $\geq$ that is complete and transitive. Let Δ^{usr} be the rank ordering of G induced by the user preference relation. Also, let Δ^{sys} be a system-provided ranking. The formulation of R_{norm} is

$$R_{\text{norm}}\left(\Delta^{\text{sys}}\right) = \frac{1}{2} \cdot \left(1 + \frac{S^{+} - S^{-}}{S^{+}{}_{\max}}\right).$$

```
demand NORMAL FORM

(APARTMENT , (at-least 1 OCCUPANTS) , (all OCCUPANTS PERSON) ,
(at-least 1 HASROOMS) , (at-least 2 HASSERVICES) , (all
HASSERVICES (SERVICES ,ROOM)), (all HASROOMS (BEDROOM ,ROOM ,
      (all HASPLACES (BED ,
            (at-least 1 OCCUPANTS) ,
            (at-most 1 OCCUPANTS) ,
            (all OCCUPANTS PERSON)) ,
            )
      (at-least 1 HASPLACES))
) , (at-most 0 HASPETS))
```

```
supply1 NORMAL FORM

(APARTMENT , (at-least 1 OCCUPANTS) , (all OCCUPANTS PERSON) ,
(at-least 1 HASROOMS) ,  (at-least 2 HASSERVICES) , (all
HASSERVICES (SERVICES , ROOM)
) , (all HASROOMS (BEDROOM ,ROOM ,
      (all HASPLACES (BED ,
            (at-least 1 OCCUPANTS) ,
            (at-most 1 OCCUPANTS) ,
            (all OCCUPANTS PERSON)) ,
            )
      (at-least 1 HASPLACES)
            (at-least 1 HASPLACES)
            (at-most 1 HASPLACES))
) (all OCCUPANTS SMOKER ,
           NONSMOKER))
```

```
supply2 NORMAL FORM

(APARTMENT , (at-least 1 OCCUPANTS) , (all OCCUPANTS PERSON) ,
(at-least 1 HASROOMS) ,  (at-least 2 HASSERVICES) , (all
HASSERVICES (SERVICES ,ROOM)
) , (at-least 2 HASROOMS) , (at-most 2 HASROOMS) , (all HASROOMS
(BEDROOM , ROOM ,
      (all HASPLACES (BED ,
            (at-least 1 OCCUPANTS) ,
            (at-most 1 OCCUPANTS) ,
            (all OCCUPANTS PERSON)) ,
            )
      (at-least 1 HASPLACES)
            (at-least 2 HASPLACES)
            (at-most 2 HASPLACES))
) (at-most 0 HASPETS))
```

```
supply3 NORMAL FORM

(APARTMENT , (at-least 1 OCCUPANTS) , (all OCCUPANTS PERSON) ,
(at-least 1 HASROOMS) , (all HASROOMS ROOM) , (at-least 2
HASSERVICES) , (all HASSERVICES (SERVICES ,ROOM)
) , (at-least 1 HASPETS))
```

```
supply4 NORMAL FORM

((at-most 1 HASSERVICES) , (all HASSERVICES BATHROOM SERVICES) ,
(at-least 2 HASPETS))
```

```
supply5 NORMAL FORM

(APARTMENT , (at-least 1 OCCUPANTS) , (all OCCUPANTS PERSON) ,
(at-least 1 HASROOMS) , (all HASROOMS ROOM) , (at-least 2
HASSERVICES) , (all HASSERVICES (SERVICES ,ROOM)
)
```

Figure 7. Normal Form for Demand and Supplies

Match	Rank
demand,supply1	0
demand,supply2	0
demand,supply3	1
demand,supply4	2
demand,supply5	0

Table 3. Ranking Results Obtained from the *rankPartial* Algorithm.

Match	Rank
demand,supply1	1
demand,supply2	0
demand,supply5	7

Table 4. Ranking Results Obtained from the *rankPotential* Algorithm.

where S^+ is the number of pairs where a better supply is ranked by the system ahead of a worse one, S^- is the number of pairs where a worse supply is ranked ahead of a better one, and S^+_{max} is the maximum possible number of S^+. It should be noted that the calculation of S^+, S^-, and S^+_{max} is based on the ranking of pairs in Δ^{sys} relative to the ranking of corresponding pairs in Δ^{usr}. R_{norm} values are in the range (0, 1). A value of 1 corresponds to a system-provided ordering of the items that is either identical to the one provided by the users or has a higher degree of resolution. Lower values correspond to a proportional disagreement between the two. As a general consideration, the system ranking was quite close to the users' rankings, and given the average volunteer orderings, the system ranking was usually in agreement with human judgment. The final averaged R_{norm} values are shown in Table 5 with reference to measures for the various types of matchings tested—that is, partial only, potential only, and mixed, both for weighted and un-weighted versions of the system.

The main gap in the system with respect to users was its use of at-least at-most restrictions, especially on price. In other words, the users evaluated proportionally the violation of the price constraint, whereas the system did not. As a result, a corrective to this issue has been introduced in the new *weighted* version of the matchmaker, by proportionally weighting numerical roles with respect to the mean value of at-least at-most restrictions. To explain better the evaluation process, the computation for a single demand/supplies matching process extracted form the experimental setting is presented below.

Let us consider the following demand: *student looking for a nice 1/2 bed flat, ch, furnished, kitchen, washing machine. Price: £150*

Also consider the following advertisements submitted as supplies:

- *Supply1: large, fully furnished room, price includes cable TV and bills. Price: £120*
- *Supply2: double room, suit a couple or two girls, required deposit. Price: £150*
- *Supply3: room to rent, suit a nonsmoking female student with worker, international preferred, deposit required. Price: £80*

Match type	R_{norm}
partial	0.73
partial weighted	0.76
potential	0.86
potential weighted	0.86
mixed	0.80
mixed weighted	0.96

Table 5. Values Obtained for R_{norm}.

Matched pairs	System ranking	Averaged user rankings
demand,supply1	2-3	2
demand,supply2	8	6
demand,supply3	4-5	4
demand,supply4	2-3	3
demand,supply5	1	1
demand,supply6	6	5
demand,supply7	4-5	8
demand,supply8	7	7

Table 6. System Results and User Preferences.

- *Supply4: single room in clean flat for nonsmoker quiet student, sharing with 2 others. Price: £120*
- *Supply5: dbl room in shared house, suit single person, use of lounge, kitchen, garden, rent includes council tax. Price: £85*
- *Supply6: female to share a room in a residential area flatshare, washing machine, TV, VCR. Price: £81*
- *Supply7: large room in family houseshare with 2 adults, suit prof / student female, viewing recommended. Price: £95*
- *Supply8: 2 bed flat, perfect for student, dble bed ADSL computer beneath. Price: £600*

The translation of these advertisements into Classic, in accordance with our ontology is pictured in Figure 8.

Table 6 shows the results from the matchmaking algorithm comparing the system-provided ranking and the average volunteer orderings for demands with respect to a supply. Figure 9 shows the same results in a graphical form.

Conclusion

This paper has presented a semantic-based matchmaking facilitator for peer-to-peer electronic marketplaces.

The definitions and properties for the matchmaking problem were devised in a framework based on description logic. The proposed classification of matches as potential and partial overcomes basic subsumption-based matchmaking, and the devised algorithms allow ranking of advertisements with respect to a given request, well modeling commonsense reasoning in

demand (and Bedroom (all toLetFor Student) (at-least 1 hasBed)
(at-most 2 hasBed) (at-least 1 hasFacilities) (all hasFacilities
(and WashingMachine NoAutonomousHeating FullyFurnished)) (all hasServices Kitchen) (all price (maximum 150)))

supply1 (and Bedroom (all price (minimum 120)) (all priceIncludes
(and Bill TVPrice)) (all hasFacilities (and FullyFurnished Spatious)))

supply2 (and (all price (minimum 150))(at-least 2 hasRoom)(at-most
2 hasRoom)(all hasRoom Room)(at-least 2 toLetFor)(at-most 2 toLetFor)(all toLetFor (and Couple Student))(at-least 1 depositRequired)(all depositRequired Yes))

supply3 (and (at-least 1 depositRequired)(all depositRequired Yes)(all price (minimum 80))
Bedroom (all toLetFor (and Student NoSmoker Worker (all sex Female))))

supply4 (and SingleRoom (all toLetFor (and NoSmoker Student))(at-least 2 occupants)(all price (minimum 120)))

supply5 (and DoubleRoom (all toLetFor Single)(all hasFacilities (and Lounge Garden))(all hasServices Kitchen)(all price (minimum 85))(all priceIncludes CouncilTax))

supply6 (and Bedroom (at-least 1 occupants) (all price (minimum 81)) (all toLetFor Female)(all hasFacilities (and WashingMachine TV VCR)))

supply7 (and Bedroom (all price (minimum 95)) (all occupants Family) (at-least 2 occupants) (all toLetFor (and Student Professional (all sex Female))))

supply8 (and Flat (at-least 2 hasBed) (at-most 2 hasBed) (all toLetFor Student) (all hasFacilities ADSL) (all price (minimum 95))))

Figure 8. Neoclassic Descriptions of Demand and Supplies

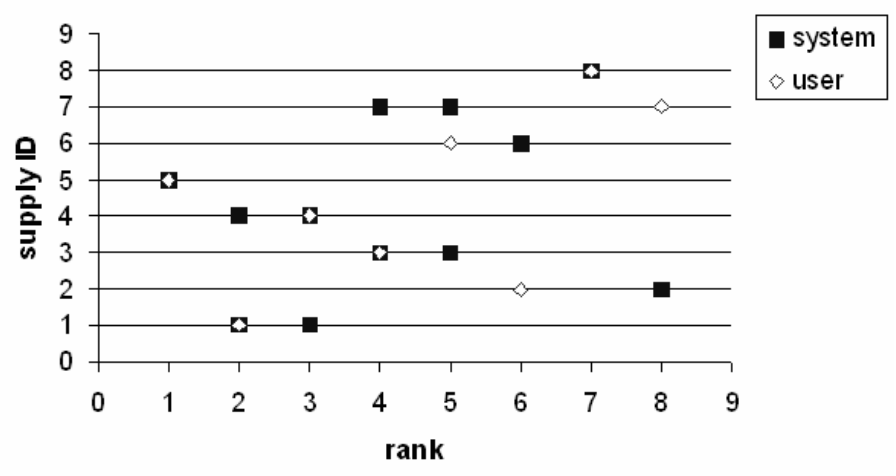

Figure 9. System and Averaged Users Ranking

the analysis of commercial advertisements. Although the theoretical framework was devised for an implementation based on the CLASSIC system, the approach retains its validity when using more expressive logics. The kind of hypothetical reasoning used in evaluating potential and partial matches also constitutes a basic motivation to study suitable theoretical frameworks and relative computational complexity for new inference services in description logics, *that is*, concept abduction and concept contraction [12, 15].

NOTES

1. Only chains of attributes are allowed as arguments to SAME-AS(.,.) in order to keep reasoning in CLASSIC decidable.
2. These services are currently implemented only for the dedicated clients.
3. A lodging was defined as neither an apartment nor a room. Advertisements were drawn from actual newspaper announcements.

REFERENCES

1. Arens, Y.; Knoblock, C.A.; and Shen, W. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems, 6* (1996), 99–130.
2. Avancha, S.; Joshi, A.; and Finin, T. Enhanced service discovery in Bluetooth. *IEEE Computer, 35,* 6 (2002), 96–99.
3. Baader, F.; Calvanese, D.; Mc Guinness, D.L.; Nardi, D.; and Patel-Schneider, P. , eds. *The Description Logic Handbook.* Cambridge: Cambridge University Press, 2002.
4. Baader, F.; Kusters, R.; Borgida, A.; and Mc Guinness, D.L. Matching in description logics. *Journal of Logic and Computation, 9,* 3 (1999), 411–447.
5. Bluetooth (www.bluetooth.com).
6. Bollmann, P.; Jochum, F.; Reiner, U.; Weissmann, V.; and Zuse, H. The LIVE-Project-Retrieval experiments based on evaluation viewpoints. In *Proceedings of the Eighth Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval*. New York: ACM Press, 1985, pp. 213–214.
7. Borgida, A. Description logics in data management. *IEEE Transactions on Knowledge and Data Engineering, 7,* 5 (1995), 671–682.
8. Borgida, A.; Brachman, R.J.; McGuinness, D.L.; and Resnick, L.A. CLASSIC: A structural data model for objects. In J. Clifford (ed.), *Proceedings of the ACM SIGMOD International Conference on Management of Data. Portland: ACM Press,* 1989, pp. 59–67.
9. Borgida, A., and Patel-Schneider, P.F. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *Journal of Artificial Intelligence Research, 1* (1994), 277–308.
10. Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 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). Seattle: ACM Press, 1998, pp. 149–158.
11. Casati, F., and Shan, M.C. Dynamic and adaptive composition of e-services. *Information Systems, 26* (2001), 143–163.
12. Colucci, S.; Di Noia, T.; Di Sciascio, E.; Donini, F.M.; and Mongiello, M. Concept abduction and contraction in description logics. In D. Calvanez (ed.), *Proceedings of the 16th International Workshop on Description Logics (DL'03),* Vol. 81. Rome: CEUR, 2003.
13. Cooke, N.J. Varieties of knowledge elicitation techniques. *International Journal of Human-Computer Studies, 41* (1994), 801–849.
14. Devambu, P.; Brachman, R.J.; Selfridge, P.J.; and Ballard, B.W. LASSIE: A knowledge-based software information system. *Communications of the ACM, 34,* 5 (1991), 36–49.
15. Di Noia, T.; Di Sciascio, E.; Donini, F.M.; and Mongiello M. Abductive matchmaking using description logics. In G. Gottlob (ed.), *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Acapulco*. Los Altos, CA: Morgan Kaufmann, 2003, pp. 337–342.
16. Di Sciascio, E.; Donini, F.M.; Mongiello, M.; and Piscitelli, G. A knowledge-based system for person-to-person e-commerce. In G. Görz (ed.), *Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001).* Vienna: CEUR, 2001.
17. Donini, F.M.; Lenzerini, M.; Nardi, D.; and Nutt, W. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall (eds.), *Proceedings of the Second International Conference on the Principles of Knowledge Representation and Reasoning (KR'91).* Los Altos, CA: Morgan Kaufmann, 1991, pp. 151–162.
18. Donini, F.M.; Lenzerini, M.; Nardi, D.; and Schaerf, A. Reasoning in description logics. In Gerhard Brewka (ed.), *Principles of Knowledge Representation: Studies in Logic, Language and Information*. Stanford: CSLI Publications, 1996, pp. 193–238.
19. Doorenbos, R.; Etzioni, O.; and Weld, D. A scalable comparison-shopping agent for the World-Wide Web. In *Proceedings of the International Conference on Autonomous Agents '97.* Marina del Rey, CA: ACM Press, 1997, pp. 39–48.
20. Euzenat, J. , ed. *Research challenges and perspectives of the Semantic Web*, October 2001 (www.ercim.org/EU-NSF/Semweb.pdf).
21. Fensel, D.; van Harmelen, F.; Horrocks, I.; McGuinness, D.; and Patel-Schneider, P.F. OIL: An ontology infrastructure for the semantic Web. *IEEE Intelligent Systems, 16,* 2 (2001), 38–45.
22. Finin, T.; Fritzson, R.; McKay, D.; and McEntire, R. KQML as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*. Gaithersburg: ACM Press, 1994, pp. 456–463.
23. Genesereth, M.R. Knowledge interchange format. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Second International Conference*. Cambridge, MA: Morgan Kauffmann, 1991, pp. 599–600.
24. Gil, Y., and Ramachandran, S. PHOSPHORUS: A task based agent

matchmaker. In *Proceedings of the International Conference on Autonomous Agents '01.* Montreal: ACM Press, 2001, pp. 110–111.
25. Gonzales-Castillo, J.; Trastour, D.; and Bartolini, C. Description logics for matchmaking of services. In G. Görz (ed.), *Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001).* Vienna: CEUR, 2001.
26. Hoffner, Y.; Facciorusso, C.; Field, S.; and Schade, A. Distribution issues in the design and implementation of a virtual market place. *Computer Networks, 32* (2000), 717–730.
27. Horrocks, I., and Sattler, U. Ontology reasoning in the SHOQ(D) description logic. In B. Nebel (ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001).* Seattle: Morgan Kauffmann, 2001, pp. 199–204.
28. Jacobs, N., and Shea, R. Carnot and Infosleuth—Database technology and the Web. In D. Schneider (ed.), *Proceedings of the ACM SIGMOD International Conference on Management of Data.* San Jose: ACM Press, 1995, pp. 443–444.
29. Jena (www.hpl.hp.com/semweb).
30. Karacapilidis, N., and Moraitis, P. Building an agent-mediated electronic commerce system with decision analysis features. *Decision Support Systems, 32* (2001), 53–69.
31. Kasbah (www.kasbah.com).
32. Knowledge representation system specification (www.bell-labs.com/user/pfps/papers/krss-spec.ps).
33. Kuokka, D., and Harada, L. Integrating information via matchmaking. *Journal of Intelligent Information Systems, 6* (1996), 261–279.
34. Li, L., and Horrocks, I. A software framework for matchmaking based on semantic Web technology. *International Journal of Electronic Commerce, 8*, 4 (2004), 39–60.
35. Maes, P.; Guttman, R.; and Moukas, A. Agents that buy and sell. *Communications of the ACM, 42,* 3 (1999), 81–91.
36. Motro, A. VAGUE: A user interface to relational databases that permits vague queries. *ACM Transactions on Office Information Systems, 6,* 3 (1988), 187–214.
37. Nebel, B. Terminological reasoning is inherently intractable. *Artificial Intelligence, 43* (1990), 235–249.
38. Paolucci, M.; Kawamura, T.; Payne, T.R.; and Sycara, K. Semantic matching of Web services capabilities. In *The Semantic Web—ISWC 2002.* Lecture Notes in Computer Science, no. 2342. New York: Springer, 2002, pp. 333–347.
39. PersonaLogic (www.PersonaLogic.com).
40. Pu, P., and Faltings, B. Enriching buyers' experience. *CHI Letters, 2,* 1 (2000), 289–296.
41. Schmid, B., and Lindemann, M. Elements of a reference model for electronic markets. In R.H. Sprague, Jr. (ed.), *Proceedings of the 31st Hawaii International Conference on System Sciences.* Los Alamitos, CA: IEEE Computer Society Press, 1998, pp. 193–201.
42. SOAP (www.w3.0rg/TR/SOAP/).
43. Ströbel, M., and Stolze, M. A matchmaking component for the discovery of agreement and negotiation spaces in electronic markets. *Group Decision and Negotiation, 11* (2002), 165–181.

44. Sycara, K.; Widoff, S.; Klusch, M.; and Lu, J. LARKS: Dynamic matchmaking among heterogeneous software agents in cyberspace. *Autonomous Agents and Multi-agent Systems, 5* (2002), 173–203.
45. Trastour, D.; Bartolini, C.; and Priest, C. Semantic Web support for the business-to-business e-commerce lifecycle. In *Proceedings of the International World Wide Web Conference (WWW) '02*. Honolulu: ACM Press, 2002, pp. 89–98.
46. Veit, D.; Muller, J.P.; Schneider, M.; and Fiehn, B. Matchmaking for autonomous agents in electronic marketplaces. In *Proceedings of the International Conference on Autonomous Agents '01.* Montreal: ACM Press, 2001, pp. 65–66.
47. Wang, H.; Liao, S.; and Liao, L. Modeling constraint-based negotiating agents. *Decision Support Systems, 33* (2002), 201–217.
48. Wright, J.R.; Weixelbaum, E.S.; Vesonder, G.T.; Brown, K.E.; Palmer, S.R.; Berman, J.I., and Moore, H.H. A knowledge-based configurator that supports sales, engineering, and manufacturing at AT&T Network Systems. *AI Magazine, 14*, 3 (1993), 69–80.

TOMMASO DI NOIA (t.dinoia@poliba.it) received a master's degree in electronic engineering from the Technical University of Bari in 2002. Currently he is a Ph.D. student in information technology engineering at the Technical University of Bari, Italy. His research interests include the application of description logics to e-commerce and knowledge management. He has published papers on these topics in various journals and in the proceedings of international conferences.

EUGENIO DI SCIASCIO (disciascio@poliba.it) received a master's degree in electronics engineering cum laude from the University of Bari in 1989 and the Ph.D. degree in 1994 from the Technical University of Bari. In 1992, he joined the University of Lecce, Italy, as an assistant professor. He is currently an associate professor of information technology engineering at the Technical University of Bari, where he is responsible for the scientific aspects of several research projects. His research interests include multimedia information retrieval, knowledge representation, and knowledge-based systems for e-commerce. He has authored or co-authored more than 80 papers on these subjects in international journals and international conferences.

FRANCESCO M. DONINI (donini@unitus.it) received his master of electronics engineering from the University of Rome—La Sapienza in 1988. He was awarded a Ph.D. degree in computer science at the same university in 1992. From 1991 to 1998, he was a researcher/assistant professor in the Department of Computer and System Science of the University of Rome—La Sapienza. Since 1998, he has been an associate professor, first at the Technical University of Bari, and currently at the University of Tuscia in Viterbo, Italy. He is co-author of many papers in international journals and international conference proceedings. The paper "Tractable Concept Languages," presented at the conference IJCAI-91 (1991), received the best paper award. He is responsible for local university research projects as well as for CNR research projects, and is editor of the area "Concept-Based Knowledge Representation" for *ETAI: Electronic Transactions on Artificial Intelligence*, published by the Royal Swedish Academy.

MARINA MONGIELLO (mongiello@poliba.it) received a master's degree in computer science cum laude from the University of Bari in 1993 and the Ph.D. degree in electronics engineering from the University of Catania in 2001. In 2002 she joined the Technical University of Bari as an assistant professor. Her research interests include knowledge-based systems for e-commerce and multimedia information retrieval. She has published several papers in international journals and conference proceedings on these topics.