A System for Principled Matchmaking in an Electronic Marketplace

Tommaso Di Noia, Eugenio Di Sciascio, Francesco M. Donini, Marina Mongiello
Politecnico di Bari, Via Re David, 200, 70125 BARI, Italy
t.dinoia,disciascio,donini,mongiello@poliba.it

ABSTRACT

More and more resources are becoming available on the Web, and there is a growing need for infrastructures that, based on advertised descriptions, are able to semantically match demands with supplies.

We formalize general properties a matchmaker should have, then we present a matchmaking facilitator, compliant with desired properties.

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 carried out show the good correspondence between users and system rankings.

Categories and Subject Descriptors

H.4.2 [**Information Systems Applications**]: Decision Support; I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages

General Terms

Algorithms, Languages, Economics

Keywords

E-commerce, Matchmaking, Knowledge Representation, Description Logics.

1. INTRODUCTION

Several models have been proposed to describe market transactions in an electronic marketplace. Schmidd and Lindemann [35] propose a classification based on four successive phases: (1) information (participants to the market seek potential partners); (2) agreement (negotiation on the terms of the agreement and agree on a contract); (3) settlement (payments and logistics); (4) after-sales (customer-support). Many researchers have concentrated their efforts 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, to begin negotiation of a business transaction, potential counterparts have to be searched and found, at least in the dynamic scenario a marketplace should be. The process of searching the space of possible matches between demands and supplies can be defined as *Matchmaking*. Notice that this process 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.

WWW2003, May 20–24, 2003, Budapest, Hungary.
ACM 1-58113-680-3/03/0005.

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 more difficult, yet challenging, when counterparts are peer entities, users or agents. As pointed out in [1] 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. 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 the 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 [15].

The uniform treatment of supplies and demands calls for technical solutions that must be 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 just on the point of view — maybe, on who is more willing to actively find the other partner.

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

Matchmaking infrastructures have been proposed in the literature (see Section 2 for a survey), but they lack a formal framework to logically classify and rank 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 framework 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 also relying on publicly available algorithms, to encourage trust, and prevent arising of doubts on fairness of the proposals returned by the matchmaking facilitator.

If supplies and demands are 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 some sort of structure in them. Such a structure could be exploited in order to evaluate "interesting" *inexact* matches. Vector-based techniques taken by classical Information Retrieval (IR) can be used, too, thus reverting matchmaking to similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker [27] or in LARKS [38]. Obviously lack of document structure in descriptions would make matching only probabilistic and strange situations may ensue, *e.g.*, consider a simple demand *"apartment with two Rooms in Soho pets allowed no smokers"* and a supply *"apartment with two Rooms in Soho, no pets, smokers allowed"*. They would correspond to a perfect match although being in obvious conflict.

Our setting allows one to categorize and rank matches according to their logical relation. In particular, we distinguish 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). Though our logical setting is basically independent from the application adopted, our infrastructure is strongly related to CLASSIC [6, 7]. CLASSIC is a knowledge representation system that, although not endowed of a language as expressive as more recent reasoners, *e.g.*, FaCT and Racer, has polynomial-time inferences and, most important, is a real system, with a rich API and concrete datatypes that can be wrapped into a host system. The infrastructure currently embeds a modified version of NeoClassic, a CLASSIC implementation, to carry out the matchmaking process.

The rest of the paper is organized as follows. In the next section we report on relevant work. Then, to make the paper self contained, we present basic notions on on DLs. In Section 4 we present a formal framework for semantic-based matchmaking, and highlight properties that a matchmaker should have. In Section 5 we propose the architecture of an infrastructure we built, implementing the devised framework. Conclusion and future work are outlined in Section 6.

Throughout the paper, we refer to an apartments rental electronic marketplace, which was chosen as case study. A subset of the ontology used as reference in the examples is reported in Figure 1.

```
 (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)

2. RELATED WORK

Though having recently become a widely investigated area, research on matchmaking can be dated back to works on vague query answering [29], where the need to overcome limitations of relational databases was addressed, with the aid of weights attributed to several search variables.

Earliest matchmakers, based on the KQML, were proposed in [16] and [27]. In these works matchmaking was introduced as an approach whereby potential producers / consumers could provide descriptions of their products/needs, either directly or through agents mediation, to be later 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 KIF (the SHADE prototype) or a free text comparison (the COINS prototype), which basically deals with descriptions as free-text retrieval tools do. Standard Information retrieval techniques have been also used in the recently proposed GRAPPA matchmaking framework [40]

Approaches similar to the cited ones were deployed in SIMS [2], which used KQML and LOOM as description language and InfoSleuth [21], which adopted KIF and the deductive database language LDL++. LOOM is also at the basis of the subsumption matching addressed in [17].

More recently there has been a growing interest towards matchmaking engines and techniques, with emphasis placed either on e-marketplaces or generic Web services, in view of the promised transformation of the Web from human understandable to the Semantic, machine understandable, Web. Significant examples include [38] and [31] where a language, LARKS, is proposed specifically designed for agent advertisement. The matching process is carried out through five progressive stages, going from classical IR analysis of text to semantic match via Θ-subsumption. The notion, inspired by Software Engineering, of *plug-in* match is introduced to overcome in some way the limitations of a matching approach based on exact match. No ranking is presented but for what is called relaxed match, which basically reverts again to a IR free-text similarity measure. So a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match.

In [39] and [18] a matchmaking framework is proposed, which operates on service descriptions in DAML+OIL and is based on the FaCT reasoner. Unfortunately as the authors admit, though very expressive, FaCT lacks of concrete datatypes, which are obviously extremely useful for, *e.g.*, e-commerce applications, and their prototype is incomplete.

Semantic service discovery via matchmaking in the Bluetooth [4] framework is investigated in [34]. Also here the issue of approximate matches, to be somehow ranked and proposed in the absence of exact matches, is discussed, but as in the previous papers no formal framework is given. Instead a logical formulation should allow to devise correct algorithms to classify and rank matches.

Also various current commercial electronic marketplaces try to provide some matchmaking capabilities between demand and supply. Jango [22] provides a system that basically only allows comparison, in terms of price, of goods available in on-line stores on the Internet. Obviously 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 [32] allows customers to impose constraints for alternatives seeking. It

must be pointed out that constraints cannot be dynamically placed but have to be taken from a pre-determined category set. Kasbah [25] is a more effective system, which allows to dynamically set constraints, yet it does not allow handling of inconsistency and partial or potential matches. A similar approach is also deployed in Tete-a-Tete [28].

A more advanced constraint based approach is proposed in [24], able to to handle conflicting preferences in demands / supplies. Consistency check of preferences is accomplished visiting an offer synthesis graph with path consistency algorithm each time a new offer is entered. A further example is Smartclient [33], 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. The underlying system basically relies on partial constraint satisfaction techniques. A recent proposal along the same lines is in [41] where negotiation agents are formally modelled using an object-oriented constraint language.

IBM's Websphere matchmaking environment is, to our knowledge, the first example of commercial solution that places an explicit emphasis on the matchmaking between a demand and a supply in a peer-to-peer way, which is referred to in [19] as *symmetric* matchmaking. As we will point 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 Corba Trading service language. Rules are basically constructed using a generic script language. Matching is then accomplished by simply comparing properties and verifying rules. No notion of distinction between full, partial potential and inconsistent matches are present.

A similar approach, with descriptions defined in XML and again a rule based decision system is in [9]. 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. In [37] an extension to the original Websphere matchmaker is proposed, which introduces users' specification of negotiable constraints when no total match is available. So the approach aims at some of the issues also addressed in this paper, but with a different, constraint based, perspective.

In [12] an initial setting for logical matchmaking was presented in a person-to-person framework.

It is noteworthy that matching in DLs has been widely treated in [3] although with no relation to matchmaking. In fact, in that work expressions denoting concepts are considered, with variables in expressions. Then a match is a substitution of variables with expressions that makes a concept expression equivalent to another. Also the more general setting of concept rewriting in DLs has no direct relation with matchmaking.

3. DESCRIPTION LOGICS AND CLASSIC SYSTEM

Description Logics are a family of logic formalisms for Knowledge Representation [5, 14]. Basic syntax elements are *concept* names, *e.g.*, `book`, `person`, `product`, `apartment`, *role* names, like `author`, `supplier`, `hasRooms` and *individuals*, such as `NewYorkCity`, `BackYardGarden`, `TVset#123`. Intuitively, concepts stand for sets of objects, and roles link objects in different concepts, 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 $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$, which consists of the *domain* Δ and the *interpretation function* $\cdot^{\mathcal{I}}$, 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.*, $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ 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 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.*, $\texttt{book} \sqcap \exists \texttt{author.italian}$ which describes the set of books whose authors include an Italian, and *universal role quantification*, *e.g.*, $\texttt{product} \sqcap \forall \texttt{supplier.japanese}$, which describes products sold only by Japanese suppliers. Other constructs may involve counting, as number restrictions: $\texttt{apartment} \sqcap (\leq\ 1\ \texttt{hasRooms})$ expresses apartments with just one room, and $\texttt{book} \sqcap (\geq\ 3\ \texttt{author})$ 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 [8].

Expressions are given a semantics by defining the interpretation function over each construct. For example, concept conjunction is interpreted as set intersection: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, and also the other boolean connectives $\sqcup$ and $\neg$, when present, are 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, $(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta \mid \forall d_2 \in \Delta : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow\ d_2 \in C^{\mathcal{I}}\}$

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 books can be divided into paperbacks and hardcover using the two inclusions $\texttt{book} \sqsubseteq \texttt{paperbacks} \sqcup \texttt{hardcover}$ and $\texttt{paperbacks} \sqsubseteq \neg\texttt{hardcover}$. Or, that books have only one title as $\texttt{book} \sqsubseteq (\leq\ 1\ \texttt{title})$. Definitions are useful to give a meaningful name to particular combinations, as in $\texttt{doubleRoom} \equiv \texttt{room} \sqcap (=\ 2\ \texttt{hasPlaces})$. Historically, sets of such inclusions are called TBox (Terminological Box). 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 $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies a definition $C = D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. 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:

1. *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?

2. *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 [6, 7] has been developed at AT&T Bell Laboratories, where it has been applied in several projects for configuration [42] and program repositories [11]. Its language has been designed with the goal to be as expressive as possible while still admitting polynomial-time inferences. So it provides intersection of concepts but no union, universal but not existential quantification over roles, and number restrictions over roles but no intersection of

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: An example KRSS SOAP packet

roles, since each of these combinations is known to make reasoning NP-hard [13]. Classic obviously provides subsumption and satisfiability reasoning services. Being a complete knowledge representation system (and not just a reasoner), Classic provides also data types as numbers and strings, and other services which are useful in a deployed prototype. Every Classic concept has an equivalent normal form, hence using such a normal form ensures syntax independence. In our current setting we use just a subset of CLASSIC constructors, *i.e.*, those needed in a $\mathcal{ALN}$ logic.

4. 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 every constraint imposed by D is fulfilled by S, hence — from the point of view of the demander — S completely satisfies D. However, note that 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, an exact match is achieved. This example already highlights the following properties that we would like to hold for a matchmaking facilitator.

PROPERTY 1 (OPEN-WORLD DESCRIPTIONS). *The absence of a characteristic in the description of a supply or demand 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.*

This property states that on one side, 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 take this issue into account.

PROPERTY 2 (NON-SYMMETRIC EVALUATION). *A matchmaking system may give different evaluations depending on whether it is trying to match a supply S with a demand D, or D with S* — i.e., *depending on who is going to use this evaluation.*

This property was evident in the above example, where all 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. Obviously such fixed-terminology problems are overcome if terms have a logical meaning through an ontology [15].

Hence, from now on we suppose that supplies and demands are expressed in a description logic DL, equipped with a model-theoretic semantics. We note that this approach includes the sets-of-keywords one, since a set of keywords can be considered also as a conjunction of concept names, *e.g.*, the set $\{apartment, soho, twoRooms\}$ can be equivalently considered as $\texttt{apartment} \sqcap \texttt{soho} \sqcap \texttt{twoRooms}$ — without modelling the structure of concepts. Obviously, entering into the structure of concepts may yield to $\texttt{apartment} \sqcap \forall \texttt{location.soho} \sqcap (= 2\, \texttt{hasRooms})$, while a keyword "noPets" can be given a logical meaning as $\forall \texttt{occupants.}(\leq 0\, \texttt{hasPets})$.

We suppose also 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 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 expressing supplies and demands, that we consider 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 been highlighted also by other researchers [39]. However, that proposal lacks a *ranking* between different potential matches, which we believe is fundamental in order to support a user in the choice of the most promising partners, among all potential ones.

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 has no potential matches, also supplies S which are inconsistent with D may be reconsidered, if the demander accepts to *revise* some of D's constraints. 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, in order to highlight the least unsatisfactory proposals, that we call *near miss* or *partial match*.

We now highlight some properties that — we believe — every ranking function should have in logical matchmaking. We state these properties in form of definitions, since we distinguish between rankings having the property from rankings that do not.

First of all, 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 1. (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 also 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, identical but for the fact that it is described to be available for June-July-August.

A similar property should hold also 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 2. (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 also for every pair of logically equivalent demands D_1, D_2 with respect to every supply S.*

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. We 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:

$$\begin{aligned} D &= \{apartment, soho, twoRooms, petsAllowed\} \\ S_1 &= \{apartment, soho, boiler, quiet\} \\ S_2 &= \{apartment, soho, boiler, quiet, lastFloor\} \end{aligned}$$

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

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

then S_3 should be ranked better than S_1 since it adds a characteristic required by D. We can generalize to concepts this example from sets, and state the following definition.

DEFINITION 3. (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, 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 of as

> *A ranking of potential matches is monotonic over subsumption if the more specific the better.*

Observe that we use the word "better" instead of using any symbol $\leq, \geq$. This is because some rankings may assume that "'better" means "increasing" (towards infinity, or 1) while others may assume "decreasing" (towards 0).

When turning to partial matches, in which some properties are already in conflict between supply and demand, the picture reverses. Now, adding another characteristic to an unsatisfactory proposal may only worsen this 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, accepting to discard one or more characteristics that we required is much worse than deciding among a ranked list of potential matches.

DEFINITION 4. (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, 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.*

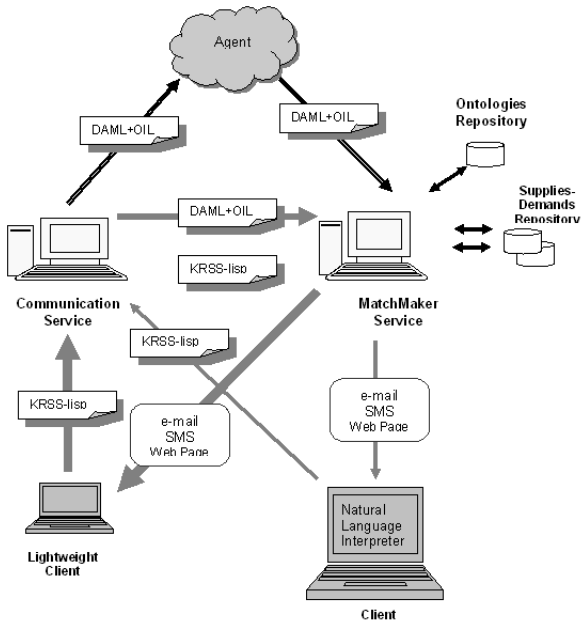

Figure 3: System Architecture of the facilitator infrastructure

Intuitively, the above property could read as: *A ranking of partial matches is antimonotonic over subsumption if the more specific, the worse.*

We remark that the properties and definitions we 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, we believe that this section keeps its significance also if one chooses more expressive DLs like $\mathcal{SHOQ}(\mathcal{D})$ [20] or even logics for which representation and reasoning system are not yet fully available, like DAML [31].

5. MATCHMAKING INFRASTRUCTURE

Using the highlighted properties as a formal specification we designed and implemented a prototype facilitator. 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 *heavy weight client* a *light weight client* and a *generic user agent*.

The heavy weight client is a Java applet that allows advertisement description in natural language. The client is anyway strongly dependent on the reference ontology, because of the inherent contextualization. Currently the only supported ontology is the apartments rental one, used in our 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) [26] of the input sentence. 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 within 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 a chart-based one. 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 our 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 light weight client is still a Java applet, but an extremely light one, in view of application on devices such as PDAs that sends, via SOAP [36], an advertisement *i.e.*, a description of the request to be matched, as a string in KRSS syntax. 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 in portable DAML+OIL ones. The module, upon receipt of the packet, transforms the string in a DAML+OIL formatted description, using Jena APIs [23]. Figure 4 shows the corresponding packet.

Advertisements can be received by the system also 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 Matchmaking service, which is the principal module of the architecture. It receives the SOAP packet, extracts the code and the URI that references the requested ontology. 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 for successful match. The same service is available for unmatched supplies. The query service, 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 [1]. For partial matches, on the basis of the matchmaker response, a demand can be revised, *e.g.*, relaxing (retracting) 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.

[1]These services are currently implemented only for the dedicated clients.

```
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

5.1 The matching engine

Our matching engine 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 returns an error message that is forwarded to the client/agent originating the request. Otherwise the proper matchmaking process takes place. To this aim we devised two algorithms, based on a modification of the original CLASSIC structural [7] subsumption algorithm, to classify and rank matches. The rationale and the structure of the algorithms are described hereafter.

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.

The algorithm easily follows a structural subsumption algorithm, except for the treatment of universal role quantification, that we explain now with the help of an example. Suppose a demand D = $\texttt{apartment} \sqcap \forall \texttt{hasRooms}.(\texttt{singleRoom} \sqcap \texttt{nonSmokerRoom})$ and two supplies C_1, C_2, defined as follows:

$$C_1 = \texttt{apartment} \sqcap \forall \texttt{hasRooms}.\texttt{roomWithTV}$$
$$C_2 = \texttt{apartment}$$

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, hence 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, $\forall \texttt{hasRooms}.\top \equiv \top$ (no restriction on the fillers of role hasRooms is equivalent to no restrictions at all). Hence, $\texttt{apartment} \equiv (\texttt{apartment} \sqcap \forall \texttt{hasRooms}.\top)$. Since we want to enforce syntax independence, both concepts should yield the same ranking. Hence, we compare the last concept, which allows us to make a recursive comparison

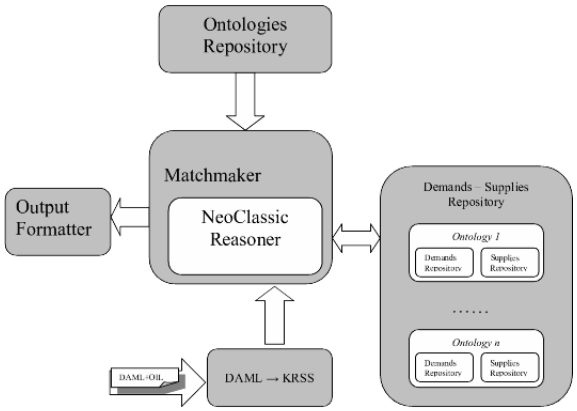

Figure 5: The architecture of the matchmaking engine

of characteristics of universal role quantifications.

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

It is easy to modify the algorithm if weights on subconcepts of D are taken into account: instead of adding 1 to n for each D's concept missing in C, one just adds 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 characteristic, or a very important one, are left unspecified in C. We implemented also a version of the algorithm 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 a universal role quantification is missing in either concept, the recursive call is unnecessary. In fact, suppose that $\forall R.E \in \mathcal{D}_{all}$ 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 $(\leq\ 0\ R)$ 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 $n \geq 0$ of C w.r.t. D,
where 0 means that $C \sqcap D$ 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 $A \in C_{names+}$
if there exists a concept $\neg A \in D_{names\neg}$
then $n := n + 1$;
/* add to n number restrictions of C */
/* which are in conflict with those of D */
for each concept $(\geq\ x\ R) \in C_\sharp$
such that there is a concept $(\leq\ y\ R) \in D_\sharp$ with $y < x$
$n := n + 1$;
for each concept $(\leq\ x\ R) \in C_\sharp$
such that there is a concept $(\geq\ y\ R) \in D_\sharp$ 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 $\forall R.F \in C_{all}$
if either (there exist both
and $\forall R.E \in D_{all}$ with $E \neq \bot$)
or ($F \neq \bot$ **and there exist both** $(\geq\ x\ R) \in D_\sharp$ with $x \geq 1$
and $\forall R.E \in D_{all}$)
then $n := n + rankPartial(F, E)$;
return n;
end algorithm

Obviously also in this case 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.

It should be noted that for both algorithms, weights can be determined according to their relevance/probability with various approaches, including, *e.g.*, utility theory based approaches [10]. In this perspective our framework can benefit of, rather than being alternative to, other methods for knowledge elicitation.

For both algorithms it can be proved they respect the properties highlighted in the previous section. With reference to complexity, it is well known [30] that the expansion of the TBox in the construction of the normal form can lead to an exponential blow-up. Nevertheless the expansion is exponential in the depth of the hierarchy of the TBox $\mathcal{T}$; if the depth of $\mathcal{T}$ is $O(\log |\mathcal{T}|)$, then the expansion is polynomial, and so also the algorithm.

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 disjoint result sets, with results ranked in the potential and partial 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 Sup-

match	rank
demand,supply1	0
demand,supply2	0
demand,supply3	1
demand,supply4	2
demand,supply5	0

Table 1: Ranking results obtained from the *rankPartial* algorithm

match	rank
demand,supply1	1
demand,supply2	0
demand,supply5	7

Table 2: Ranking results obtained from the *rankPotential* algorithm

plies 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. To simplify the reading no weights are used in the following.

5.1.1 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*[2]; *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 1 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 w.r.t. the supplies, *i.e.*, how much should be revised in the demand. The system can also explicit unsatisfied concepts of the demand. In this way the user can decide to revise his/her demand advertisement.

Remaining supplies are all *potential* matches, *i.e.*, there is nothing in conflict with demand in them. Yet it is obvious that a user needs to know those supplies that may best match the demand. The execution of *rankPotential* algorithm provides the answer. Results in Table 2 show that supply2 is an exact match w.r.t. the given demand. Supplies 1 and 5 are potential matches, *i.e.*, there is no characteristic in conflict but some of the requests in demand are not explicitly available. The user may then, *e.g.*, when no exact match is available, carry out further investigations on potential matches. In this process he/she is again helped by the system ranking, which represents the distance between the demand and supply.

It is noteworthy that the ranking also prevents advertisements

[2]A lodging is considered as neither an apartment nor a room; advertisements were drawn from actual newspapers announcements.

```
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

from being submitted in an extremely generic way. Simple subsumption matching [39] 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.

The non-symmetric behavior of the matchmaking process can also be highlighted here by simply exchanging the arguments of the matchmaking algorithms. By taking, *e.g.*, the description of supply1 and matching it against the example demand, the *rankPotential* algorithm returns a rank 4.

5.1.2 Experiments

We believe that degree of conformance of an automated matchmaking facilitator to users' perception is of extreme importance, especially in a setting as our own, which ranks and categorizes matches.

To evaluate the ability of the system to meet users' expectations we set up a little experiment.

We selected all apartments-rental advertisements from the dedicated section of a local newspaper on Oct. 6th, 2002. We subdivided them in two sets, *demands* (23 advertisements) and *supplies* (39 advertisements). We submitted to twenty volunteers of various sex and age, a questionnaire that included 12 items. Each item was one demand and a set of up to eight supplies, or one supply and a set of up to eight demands. Volunteers were asked to rank, according to their judgement, 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). Volunteers were given unlimited time and in average it took approximately half hour to complete the questionnaire.

Groups of advertisements where chosen by the authors so that items included only potential matches, only partial matches, and both potential and partial matches. It is noteworthy that, at least for the single-day pick we made, we were not able to detect any exact match. After that the same sets of items were submitted to the matchmaking engine.

As a general consideration, the system response was quite close to the users' ones, and considering average volunteers orderings the systems rankings was in agreement with the human judgement almost always. Standard deviation w.r.t. averaged users ranking was:

$\sigma_{potentialmatches} = 8.3\%$, $\sigma_{partialmatches} = 9.7\%$, $\sigma_{mixed} = 9.1\%$.

Figure 8 shows a single result, in a graphical form, for mixed potential and partial demands w.r.t. to a supply. Notice that, while volunteers gave strictly ranked orderings, the system could also provide equal ranking for various matches. It should be noticed that the main gap the system had 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, while the system did not. We are hence introducing in the *weighted* version of the matchmaker a corrective to this issue, by proportionally weighting numerical roles w.r.t. the mean value of at-least at-most restrictions.

```
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

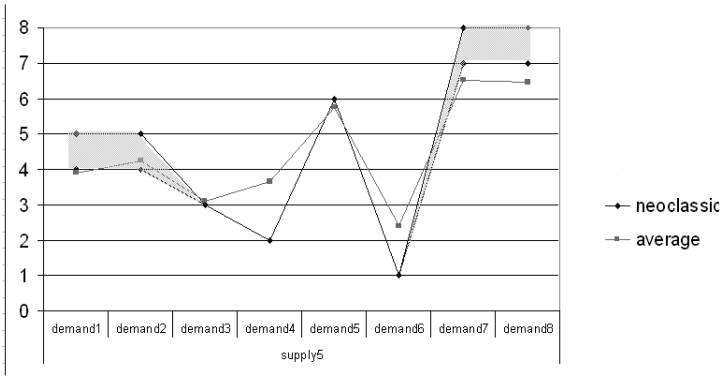

Figure 8: System and averaged users ranking

6. CONCLUSION

In this paper we have proposed a semantic based matchmaking facilitator for peer-to-peer electronic marketplaces.

The core engine has been implemented modifying the NeoClassic subsumption algorithm in a way that allows categorization of match type and the ranking of matches. While large scale experiments are in progress, a simple experiment with a case study ontology has shown a good correspondence with human perception of the system provided rankings.

We are currently working on an extension of the system to DAML-S and UDDI, which will also feature a negotiation module. An application of the current approach to Bluetooth, with a stack modified to incorporate semantic descriptions along the lines highlighted in [34], has also been implemented and is being tested.

The kind of hypothetical reasoning used in evaluating potential matches gave us also the basic motivation to study abduction in DLs and relative computational complexity. A suitable theoretical framework has been devised and first results are forthcoming.

7. ACKNOWLEDGMENTS

We thank Peter Patel-Schneider for valuable support on CLASSICsystem.

This work has been supported by MURST project CLUSTER22, by EU-POP project *"Negotiation Agents for the Electronic Marketplace"*, by project *''Tecnologie innovative per la valorizzazione e la fruizione dei Beni Culturali"*, by Italian CNR projects LAICO, and *"Metodi di Ragionamento Automatico nella modellazione ed analisi di dominio"*.

8. REFERENCES

[1] Research challenges and perspectives of the Semantic Web. In *Report of the EU-NSF Strategic Workshop*, October 2001. http://www.ercim.org/EU-NSF/Semweb.pdf.

[2] Y. Arens, C. A. Knoblock, and W. Shen. Query Reformulation for Dynamic Information Integration. *J. of Intelligent Information Systems*, 6:99–130, 1996.

[3] F. Baader, R. Kusters, A. Borgida, and D. Mc Guinness. Matching in Description Logics. *J. of Log. and Comp.*, 9(3):411–447, 1999.

[4] Bluetooth. http://www.bluetooth.com.

[5] A. Borgida. Description Logics in Data Management. *IEEE TKDE*, 7(5):671–682, 1995.

[6] A. Borgida, R. J. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A Structural Data Model for Objects. In *Proc. of ACM SIGMOD*, pages 59–67, 1989.

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

[8] D. Calvanese, G. De Giacomo, and M. Lenzerini. On the Decidability of Query Containment under Constraints. In *Proc. of PODS'98*, pages 149–158, 1998.

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

[10] N. J. Cooke. Varieties of Knowledge Elicitation Techniques. *Intl. Journal of Human-Computer Studies*, 41:801–849, 1994.

[11] P. Devambu, R. J. Brachman, P. J. Selfridge, and B. W. Ballard. LASSIE: A Knowledge-Based Software Information System. *Comm. of the ACM*, 34(5):36–49, 1991.

[12] 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.

[13] F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The Complexity of Concept Languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proc. of KR'91*, pages 151–162. Morgan Kaufmann, Los Altos,CA, 1991.

[14] 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.

[15] 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.

[16] T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proc. of CIKM'94*, pages 456–463. ACM, 1994.

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

[18] J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of Workshop on Application of Description Logics*, September 2001.

[19] 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.

[20] I. Horrocks and U. Sattler. Ontology Reasoning in the SHOQ(D) Description Logic. In *Proc. of IJCAI 2001*, pages 199–204, 2001.

[21] N. Jacobs and R. Shea. Carnot and Infosleuth – Database Technology and the Web. In *Proc. of ACM SIGMOD*, pages 443–444. ACM, 1995.

[22] Jango. http://www.Jango.com.

[23] Jena. http://www.hpl.hp.com/semweb/.

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

[25] Kasbah. http://www.kasbah.com.

[26] Knowledge Representation System Specification. http://www.bell-labs.com/user/pfps/papers/krss-spec.ps.

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

[28] P. Maes, R. Guttman, and A. Moukas. Agents that Buy and Sell. *Comm. of the ACM*, 42(3):81–91, 1999.

[29] A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM TOOIS*, 6(3):187–214, 1988.

[30] B. Nebel. Terminological Reasoning is Inherently Intractable. *Artif. Intell.*, 43:235–249, 1990.

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

[32] PersonaLogic. http://www.PersonaLogic.com.

[33] P. Pu and B. Faltings. Enriching Buyers' Experience. *CHI Letters*, 2(1), 2000.

[34] S. Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, pages 96–99, 2002.

[35] B. Schmidd and M. Lindemann. Elements of a Reference Model for Electronic Markets. In *Proc. of the 31st HICSS Conference*, 1998.

[36] SOAP. http://www.w3.org/TR/SOAP/.

[37] M. Ströbel and M. Stolze. A Matchmaking Component for the Discovery of Agreement and Negotiation Spaces in Electronic Markets. *Group Decision and Negotiation*, 11:165–181, 2002.

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

[39] 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.

[40] D. Veit, J. Muller, M. Schneider, and B. Fiehn. Matchmaking for Autonomous Agents in Electronic Marketplaces. In *Proc. AGENTS '01*, pages 65–66. ACM, 2001.

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

[42] J. R. Wright, E. S. Weixelbaum, G. T. Vesonder, K. E. Brown, S. R. Palmer, J. I. Berman, and H. H. Moore. A Knowledge-Based Configurator that Supports Sales, Engineering, and Manufacturing at AT&T Network Systems. *AI Magazine*, 14(3):69–80, 1993.