

Vague Knowledge Bases for Matchmaking in P2P E-Marketplaces

Azzurra Ragone¹, Umberto Straccia² Tommaso Di Noia¹,
Eugenio Di Sciascio¹, Francesco M. Donini³

¹ SisInf Lab - Politecnico di Bari, Via Re David, 200, I-70125, Bari, Italy
{a.ragone,t.dinoia,disciascio}@poliba.it

² ISTI - CNR, Via G. Moruzzi 1, 56124 Pisa, Italy
straccia@isti.cnr.it

³ Università della Tuscia, via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

Abstract. In this paper we propose an approach to semantic matchmaking that exploits various knowledge representation technologies to find most promising partners in peer-to-peer e-marketplaces. In particular we mix in a formal and principled way the semantic expressiveness of DLR-lite Logic Programs, fuzzy logic and utility theory. We adopt DLR-Lite Logic Programs to obtain a reasonable compromise between expressiveness and complexity to ensure the scalability of our approach to large e-marketplaces, and Fuzzy Logic to model logical specifications as soft constraints. Furthermore, fully exploiting the peer-to-peer paradigm, we consider in the matchmaking process preferences and corresponding utilities of both parties.

1 Introduction

The distinguishing characteristic of a peer-to-peer (P2P) e-marketplace is that basically peer users – both buyers and sellers – can submit their advertisements, browse through available ads, and be assisted in finding the best available counterparts to meet their needs and initiate a commercial transaction. Furthermore, in such marketplaces, there is often the need to negotiate not only on single numerical features such as price, quantity, etc., but also on some good’s characteristics. Then there is the need to represent advertisements in a machine understandable way, using languages able to model the background domain knowledge. Also, descriptions could have logical implications, *e.g.*, *If a car has leather seats then it is also provided with air conditioning* or bundles *e.g.*, *Sports car with optional package including both GPS system and alarm system*, and some kind of logical theory, able to let users express their needs/offers, could surely help. Finally, while performing a matchmaking process between two peers advertisements, we should take into account user preferences – soft constraints – and distinguish them from mandatory – hard – ones.

In an e-commerce setting, matchmaking can be defined as the process of finding “good” counterparts for a given entry in the marketplace. Of course, the evaluation of how “good” a counterpart is constitutes most of the effectiveness of a matchmaking system. Currently, many commercial sites force the buyer to enter her request browsing

a predefined classification that may be completely unsuitable for the characteristics the buyer might have in mind *e.g.*, they require to enter a brand first, then a model of that brand, etc. while a buyer may be not interested in a specific brand, but only on some limitations on price and color. In this respect, one may say that they provide no matchmaking assistance: the matchmaker is the buyer herself. Even the use of textual search engine (as in eBay) does not help a lot, since the result is a (sometimes very long and tedious) list to browse.

To assist buyers and sellers in marketplaces, several research proposals on matchmaking systems were issued. They either try to compute a score of possible counterparts, based on textual information [25], or to compare the logical representations of supply and demand [10], or combine both scores and logic in some way [7, 16]. Our proposal falls in this last category, mixing in a formal way ontologies in DLR-Lite, Datalog rules, Fuzzy sets, and Utility Theory. While all the above logic-based proposals suffer on the scalability side, our resort on DLR-Lite and Datalog ensures the scalability of our approach w.r.t. to large datasets.

Furthermore, following the economical approach to negotiation, the matchmaker computes a score as the maximum value of the *product* of the weighted utility of the buyer u_β times the weighted utility of the seller u_σ over all possible agreements between the buyer and the seller. In this way both buyer's and seller's preferences are taken into account ruling out of the match list those counteroffers that, although seemingly appealing for the buyer, would probably lead to failure due to contrasting preferences of the seller, that we take already into account.

The remaining of the paper is as follows: Section 2 introduces basics of languages and technologies we adopt. In Section 3 requirements for matchmaking process, including preferences, utility functions are illustrated and vague knowledge bases are introduced. The description of the matchmaking process over vague knowledge bases follows in Section 4. Next, in Section 5 rules for item classification in P2P marketplaces are outlined. An illustrative example is presented in Section 6, discussion about relevant related work and conclusions close the paper.

2 Basic technologies

As our representation and query tool, we use a specific combination of Description Logics [3] (DLs), Logic Programming [6] (LPs) and Fuzzy Theory [26].

A *Knowledge Base* is a triple $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$, where \mathcal{F} is a set of facts stored into a relational database, \mathcal{O} is the DL component and \mathcal{P} is the rule component. The DL component is used to model the application domain's ontology (*e.g.*, the intentional knowledge). Specifically, we use a description logic of the family of DLs, *DLR-Lite* [5]. Concerning the rule and query component, we use an extension of *Datalog* (cf. [6] among others), in which we allow soft constraint predicates to appear in rules and queries [22]. Basically, we allow vague/fuzzy predicates to occur in rule bodies, which have the effect that each tuple in the answer set of a query has now a score in [0,1]. The main problem to be addressed in the resulting language is the problem to compute the top-k answers in case the set of facts is huge, without evaluating all the

tuples' score. As matching a buyer's request with a seller's offer is a matter of degree, our purpose is to find the top- k matches only, rather than all matches.

In the following, consider a knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$.

Facts component. Concerning the facts component, \mathcal{F} is simply a finite set of formulae of the form $R(c_1, \dots, c_n)$, where R is an n -ary predicate and c_i are constants. Facts, representing extensional information, are stored in relational tables of an underlying database. For instance,

$$CarTable(544, FiatPunto, 2004, 15000)$$

is a fact stating that item 544 is a Fiat Punto, built in 2004 and having 15000 kilometers.

An *interpretation* $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ consists of a *fixed infinite domain* Δ and an *interpretation function* $\cdot^{\mathcal{I}}$ that maps an n -ary predicate R into an n -ary relation $R^{\mathcal{I}}$ over Δ and maps constants into constants of Δ such that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$ (unique name assumption). We assume to have one object for each constant, denoting exactly that object. In other words, we have standard names, and we will not distinguish between the alphabet of constants and the objects in Δ .

DL component. Concerning the DL component, \mathcal{O} is a finite set of DLR-Lite axioms⁴. A DLR-Lite *axiom* has the form $C_1 \sqsubseteq C_2$ (*concept inclusion*) or has the form $(disjoint\ C_1, \dots, C_n)$ (*disjointness axiom*), where C_i is a concept expression. Informally, $C_1 \sqsubseteq C_2$ says that the set denoted by C_1 is a subset of the set denoted by C_2 , while $(disjoint\ C_1, \dots, C_n)$ declares that the concepts are pairwise disjoint. Concepts expressions are constructed starting from a set of atomic concepts and relations by applying suitable constructs. In DLR-Lite we distinguish between constructs that are allowed in the left-hand side (Cl) and those in the right-hand side (Cr) of concept inclusions, according the following syntax:

$$\begin{aligned} Cl &\longrightarrow A \mid \exists i : R \mid Cl_1 \sqcup Cl_2 \\ Cr &\longrightarrow A \mid \exists i : R \mid Cr_1 \sqcap Cr_2 \end{aligned}$$

where R is an n -ary predicate and $i \in \{1, \dots, n\}$.

An interpretation $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ maps a concept C into subsets $C^{\mathcal{I}}$ of Δ . In the following, R denotes an n -ary predicate, and we use \mathbf{c} to denote an n -tuple of constants, and $\mathbf{c}[i]$ to denote the i -th component of \mathbf{c} . Then $\cdot^{\mathcal{I}}$ has to satisfy:

$$\begin{aligned} A^{\mathcal{I}} &\subseteq \Delta \\ (C_1 \sqcap C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}} \\ (C_1 \sqcup C_2)^{\mathcal{I}} &= C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}} \\ (\exists i : R)^{\mathcal{I}} &= \{\mathbf{c}[i] \mid \mathbf{c} \in R^{\mathcal{I}}\} \end{aligned}$$

An interpretation *satisfies (is a models of)* $C_1 \sqsubseteq C_2$ iff $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, while an interpretation *satisfies (is a models of)* $(disjoint\ C_1, \dots, C_n)$ iff $\forall i \neq j. C_i^{\mathcal{I}} \cap C_j^{\mathcal{I}} = \emptyset$.

⁴ Here we refer to DLR-Lite_{core} but any DL of the DL-Lite or DLR-Lite family can be as well considered.

Example 1. The following set of axioms is an excerpt of the encoding for the web directory behind the car selling site www.autos.com:

$$\begin{array}{ll}
Cars \sqsubseteq Vehicles & CompactCars \sqsubseteq PassengerCars \\
Trucks \sqsubseteq Vehicles & Vehicles \sqsubseteq \exists 1 : hasMaker \sqcap \exists 1 : hasPrice \\
Vans \sqsubseteq Vehicles & \exists 1 : hasPrice \sqcup \exists 1 : hasMaker \sqsubseteq Vehicles \\
LuxuryCars \sqsubseteq Cars & \exists 2 : hasMaker \sqsubseteq CarMaker \\
PassengerCars \sqsubseteq Cars & Cars \sqsubseteq \exists 1 : hasKmWarranty \sqcap \exists 1 : hasFuel \\
(disjoint\ Cars, Trucks, Vans) & \exists 2 : hasFuel \sqsubseteq FuelType
\end{array}$$

Given a DLR-Lite ontology related to a relation model, in [5] it is shown how to rewrite a conceptual query over the ontological model in a conjunctive query over the relational model in the form

$$q(\mathbf{x}) \leftarrow \exists \mathbf{y} body(\mathbf{x}, \mathbf{y})$$

Here *body* is the conjunction of n -ary predicates representing the information modeled by concepts and roles in the DLR-Lite ontology.

Notice that even apparently simple, DLR-Lite family languages are expressive enough to represent RDFS⁵ ontologies (at least their DL subset).

LP component. Concerning the rule component of a knowledge base, \mathcal{P} is a finite set of vague Datalog rules[22], which are defined as follows.

A Datalog *rule* is a Horn clause of the form

$$P(\mathbf{t}_0) \leftarrow R_1(\mathbf{t}_1), \dots, R_n(\mathbf{t}_n),$$

where $P(\mathbf{t}_0)$ is the *head* of the rule, and $R_1(\mathbf{t}_1), \dots, R_n(\mathbf{t}_n)$ is the *body* of the rule. $P(\mathbf{t}_0)$ and all $R_i(\mathbf{t}_i)$ are atoms, \mathbf{t}_i are arrays of *terms*. A *term* is either a *variable* or a *constant*. Here we also provide a set of built-in predicates, like $+$, $*$, $-$, $/$, \geq , \leq , $=$, with (obvious) fixed interpretation, which may appear in a rule body.

A Datalog *query predicate* q is a designated n -ary predicate symbol appearing in the head of a Datalog rule in \mathcal{P} . For instance,

$$q(x, p) \leftarrow Cars(x), hasPrice(x, p), p \leq 15000$$

is a query asking for cars whose price is less or equal than 15000.

The interpretation of n -ary predicates, terms and the notions of satisfiability (is model of) and logical consequence are as usual.

Vague Datalog rules are as Datalog rules except that we additionally allow fuzzy predicates to occur in Datalog rule bodies (see [22]). Specifically, let r be an n -ary predicate symbol. We add an additional position to r , making it $n + 1$ -ary. Then a vague Datalog *rule* is of the form

$$r(\mathbf{x}, s) \leftarrow \exists \mathbf{y} body(\mathbf{x}, \mathbf{y}), s = f(p_1(\mathbf{z}_1), \dots, p_n(\mathbf{z}_n))$$

where

1. \mathbf{x} are the n distinguished variables;

⁵ <http://www.w3.org/TR/rfs-schema/>

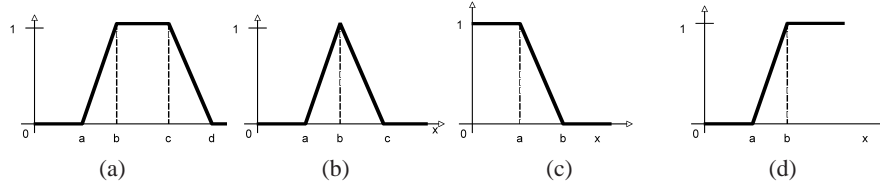


Fig. 1. (a) Trapezoidal function; (b) Triangular function; (c) L -function; (d) R -function

2. s is the *score variable*, taking values in $[0, 1]$, and r is functional on s ;
3. \mathbf{y} are so-called *non-distinguished variables* and are distinct from the variables in \mathbf{x} ;
4. $body(\mathbf{x}, \mathbf{y})$ is a conjunction of Datalog atoms;
5. \mathbf{z}_i are tuples of constants or variables in \mathbf{x} or \mathbf{y} ;
6. p_i is an n_i -ary *fuzzy predicate* assigning to each n_i -ary tuple \mathbf{c}_i as score $p_i(\mathbf{c}_i) \in [0, 1]$;
7. f is a *scoring function* $f: [0, 1]^n \rightarrow [0, 1]$, which combines the scores of the n fuzzy predicates p_i into an overall *query score* to be assigned to the score variable s . We assume that f is *monotone*, i.e., for each $\mathbf{v}, \mathbf{v}' \in [0, 1]^n$ such that $\mathbf{v} \leq \mathbf{v}'$, $f(\mathbf{v}) \leq f(\mathbf{v}')$ holds, where $(v_1, \dots, v_n) \leq (v'_1, \dots, v'_n)$ iff $v_i \leq v'_i$ for all i ; we assume that the computational cost of f and all fuzzy predicates p_i is bounded by a constant.

We call $s = f(p_1(\mathbf{z}_1), \dots, p_n(\mathbf{z}_n))$ a *scoring atom*. For instance,

$$\begin{aligned} CheapCar(x, p, s) &\leftarrow NewCar(x), hasPrice(x, p), \\ &\quad s = \max(0, 1 - p/15000) \\ CheapCar(x, p, s) &\leftarrow SecondHandCar(x), hasPrice(x, p), \\ &\quad s = \max(0, 1 - p/7500) \end{aligned}$$

are vague Datalog rules that can be used to look for cheap cars, assigning to each car a score depending on its price. If the price of a new car is above 15,000€ the car is not considered as a cheap one, while the scoring function is increased as the price lowers. Hence, it is quite natural that if we are looking for cheap cars one wants that the retrieved cars are sorted in decreasing order with respect to its score, i.e., degree of cheapness. Furthermore, as the database may contain thousands of tuples, one usually wants to retrieve just the top- k ranked ones.

Concerning fuzzy predicates involved in scoring atoms, we recall that in fuzzy set theory and practice there are many membership functions for fuzzy sets membership specification. However, the *trapezoidal* $trz(x; k_1, k_2, a, b, c, d)$, the *triangular* $tri(x; k_1, k_2, a, b, c)$, the *L-function* (left shoulder function) $LS(x; k_1, k_2, a, b)$ and the *R-function* (right shoulder function) $RS(x; k_1, k_2, a, b)$ are simple, yet most frequently used to specify membership degrees (see Figure 1) ⁶.

From a semantics point of view (see [22]), we have to take into account the additional scoring atom $s = f(p_1(\mathbf{z}_1), \dots, p_n(\mathbf{z}_n))$.

⁶ k_1, k_2 is the domain of the functions.

Informally a vague Datalog rule is interpreted in an interpretation \mathcal{I} as the set $r^{\mathcal{I}}$ of tuples $\langle \mathbf{c}, v \rangle$, such that when we substitute the variables \mathbf{x} and s with the constants \mathbf{c} and the score value $v \in [0, 1]$, the formula $\exists \mathbf{y} \text{body}(\mathbf{x}, \mathbf{y}), s = f(p_1(\mathbf{z}_1), \dots, p_n(\mathbf{z}_n))$ evaluates to true in \mathcal{I} .

Due to the existential quantification $\exists \mathbf{y}$, for a fixed \mathbf{c} , there may be many substitutions \mathbf{c}' for \mathbf{y} and, thus, we may have many possible scores for the tuple \mathbf{c} . Among all these scores for \mathbf{c} , we select the highest one, *i.e.*, the sup.

In case that the atom $r(\mathbf{x}, s)$ is the head of multiple rules, for each tuple \mathbf{c} there may be a score v_i computed by each of these rules. In that case, we assume that the overall score for \mathbf{c} is the *maximum* among the scores v_i .

Now, let $\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v} = \{\mathbf{x}/\mathbf{c}, \mathbf{y}/\mathbf{c}', s/v\}$ be a substitution of the variables \mathbf{x}, \mathbf{y} and s with the tuples \mathbf{c}, \mathbf{c}' and score value $v \in [0, 1]$. Let $\psi(\mathbf{x}, \mathbf{y}, s)$ be $\text{body}(\mathbf{x}, \mathbf{y}), s = f(p_1(\mathbf{z}_1), \dots, p_n(\mathbf{z}_n))$. With $\psi(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ we denote the ground formula obtained by applying the substitution $\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ to $\psi(\mathbf{x}, \mathbf{y}, s)$.

We say that an interpretation \mathcal{I} is a *model* of $\psi(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ iff $\psi(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ evaluates to true in \mathcal{I} , *i.e.*, all ground atoms and the grounded scoring atom occurring in $\psi(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ are true. We will write $\mathcal{I} \models \psi(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v}$ in this case.

Then, the interpretation $\mathbf{r}^{\mathcal{I}}$ of a set of rules with same head $\mathbf{r} = \{r_1, \dots, r_n\}$ in \mathcal{I} is

$$\mathbf{r}^{\mathcal{I}} = \{\langle \mathbf{c}, v \rangle \mid v = \max(v_1, \dots, v_n), v_i = \sup_{\mathbf{c}'} \{v' \mid \mathcal{I} \models \psi_i(\mathbf{x}, \mathbf{y}, s)\theta_{\mathbf{x}\mathbf{y}s}^{\mathbf{c}\mathbf{c}'v'}\}\}, \quad (1)$$

where each rule $r_i \in \mathbf{r}$ is of the form $r(\mathbf{x}, s) \leftarrow \exists \mathbf{y} \psi_i(\mathbf{x}, \mathbf{y}, s)$, $\sup \emptyset$ is undefined, and $\max(v_1, \dots, v_n)$ is undefined iff all its arguments are undefined.

Note that some tuples \mathbf{c} may not have a score in \mathcal{I} and, thus, $\langle \mathbf{c}, v \rangle \notin \mathbf{r}^{\mathcal{I}}$ for no $v \in [0, 1]$. Alternatively we may define $\sup \emptyset = 0$ and, thus, all tuples \mathbf{c} have a score in \mathcal{I} , *i.e.*, $\langle \mathbf{c}, v \rangle \in \mathbf{r}^{\mathcal{I}}$ for some $v \in [0, 1]$. We use the former formulation to distinguish the case where a tuple \mathbf{c} is retrieved, though the score is 0, from the tuples which do not satisfy the query and, thus, are not retrieved. Finally, for all \mathbf{c} and for all $v \in [0, 1]$, we say that \mathcal{I} is a *model* of $r(\mathbf{c}, v)$ (denoted $\mathcal{I} \models r(\mathbf{c}, v)$) iff $\langle \mathbf{c}, v \rangle \in \mathbf{r}^{\mathcal{I}}$.

We say that a vague knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$ *entails* $q(\mathbf{c}, v)$, written $\mathcal{K} \models q(\mathbf{c}, v)$, iff for all models \mathcal{I} of \mathcal{K} , $\mathcal{I} \models q(\mathbf{c}, v)$ holds.

Linking **Facts component** \mathcal{F} , **DL component** \mathcal{O} and **LP component** \mathcal{P} with each other in \mathcal{K} we have that:

- Atoms, representing unary predicates, and predicates occurring in \mathcal{O} may appear in rules in \mathcal{P} .
- Predicates occurring in \mathcal{F} do not occur in the head of rules in \mathcal{P} — essentially, we do not allow that the fact predicates occurring in \mathcal{F} can be redefined by \mathcal{P} .

2.1 Top- k Retrieval

The basic inference services that concerns us is the top- k retrieval problem, where this latter is defined as:

Top- k retrieval: Given a vague knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$, retrieve the top- k ranked tuples $\langle \mathbf{c}, v \rangle$ that instantiate the query q and rank them in decreasing order w.r.t.

the score v , *i.e.*, find the top- k ranked tuples of the answer set of q , denoted

$$ans_k(\mathcal{K}, q) = \text{Top}_k\{\langle \mathbf{c}, v \rangle \mid \mathcal{K} \models q(\mathbf{c}, v)\}.$$

For instance,

$$q(x, p, s) \leftarrow \text{CheapCar}(x, p, s)$$

is a query asking for cheap cars. The top- k ranked cars, according to the score (that depends on their price), is obtained by $ans_k(\mathcal{K}, q)$.

From a reasoning point of view, [23] shows that the top- k problem for DL-Lite knowledge bases can be solved in LogSpace data complexity. The result holds also for the top- k problem in DLR-Lite, using the results described in [5]. On the other hand, [22] shows that the top- k problem for vague Datalog can also be solved in LogSpace data complexity, if the set of vague Datalog rules is not recursive. Both solutions rely on a query rewriting method which is conceptually based on the same idea as for [5]. In fact, it can be shown that for a vague knowledge base $\mathcal{K} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$, where \mathcal{P} is not recursive, the top- k problem can be solved in LogSpace data complexity. Essentially, we combine [22] and [23] to rewrite a query into a set R of new queries and then we apply relational top- k database technology (see, *e.g.*, [14]) to solve the queries in R .

3 Matchmaking Scenario

In this section we outline the matchmaking scenario and show how to model it as a top- k retrieval problem over a vague knowledge base \mathcal{K} . Without loss of generality, in the examples we refer to an automobile marketplace, and motivate our work in this domain. In such domain features as look, comfort, optionals, type have to be modeled, as well as numerical features as price, warranty or delivery time. In fact, based on these features both the buyer is able to formulate her request and the seller to describe the good to be sold *i.e.*, a buyer can specify conditional preferences, such as “**If** it is a luxury car, **then** it has to be provided with leather seats” or “I want a cheap car, yet if the car has an alarm system I’m ready to pay up to 18,000 €”, conversely the seller can offer “a compact car with 4 years warranty or 12,000 km warranty”. Constraints can involve only numerical features, or non numerical ones, as well as both of them.

Hard and Soft Constraints. In a typical e-marketplace scenario, the issues within both the buyer’s request and the seller’s offer can be split into *strict requirements* and *preferences*. Strict requirements represent what the buyer and the seller want to be necessarily satisfied in order to accept the final agreement – in our framework we call strict requirements *hard constraints*. Preferences denote issues they are willing to negotiate on – this is what we call *soft constraints*. Hence, the matchmaker has to be able to handle both hard and soft specifications of both the buyer and the seller. Let us now introduce an example request, we will use to explain some aspects of our approach:

Example 2. Suppose to have a buyer’s request like “I want a passenger car black or gray. Preferably I would like to pay less than 14,000 € furthermore I’m willing to pay up to 17,000 € if warranty is greater or equal than 100,000 km.”. In this example we identify:

Hard Constraints. *Body Type:* Passenger car; *Color:* Black or Grey.

Soft Constraints. *Price:* $\leq 14,000$; *Warranty-Price:* if Warranty $\geq 100,000$ then Price $\leq 17,000$ €.

Utility. Given a request and several supplies, in the final agreement, both buyer's and seller's *hard constraints* have to be satisfied. Nevertheless, how should the matchmaker find –and rank– the most *suitable* or *promising* agreements to be proposed to both parties? In the ranking process, *soft constraints* are key information the matchmaker should use to evaluate the match degree. The final agreement is computed in order to maximize both buyer's and seller's preferences satisfaction. For instance, w.r.t. Example 2 suppose to have three supplies⁷:

σ' = *Body Type:* Compact car; *Color:* Grey; *Price:* 16,000 €; *Warranty:* 200,000 km.

σ'' = *Body Type:* Passenger Car; *Color:* Black; *Price:* 13,000 €; *Warranty:* 50,000 km.

σ''' = *Body Type:* Van; *Color:* Brown; *Price:* 19,000 €.

Comparing these supplies with buyer's request we note that σ''' will be discarded because its hard constraints are in conflict with the buyer's one. For σ' and σ'' we have that: σ' satisfies the preference $\beta_2 = \{\mathbf{Warranty-Price:}$ if Warranty $\geq 100,000$ then Price $\leq 17,000$ € $\}$; σ'' the preference $\beta_1 = \{\mathbf{Price:}$ $\leq 14,000\}$. Now the question is: *how to evaluate the best one?*

In order to provide an answer to such a question, we take into account utility values assigned by the buyer and representing the preference relevance to sub-parts of *soft constraints*. In this case we assume utility values — $u(\beta_1)$ and $u(\beta_2)$ — both for β_1 and β_2 .⁸ Notice that actually the same holds from the seller's side. In a P2P e-marketplace the seller may express his preferences — *soft constraints e.g.*, on selling price, warranty, delivery time — with corresponding utilities $u(\sigma_j)$, as well as his *hard constraints* (e.g., color, model, engine fuel, etc.). The only constraint on utility values is that both seller's and buyer's ones are normalized to 1 to eliminate outliers, and make them comparable [12].

$$\sum u(\beta_i) = 1, \sum u(\sigma_j) = 1 \quad (2)$$

Since we assume utilities on preferences as additive, here we can write the global utility of the buyer u_β and of the seller u_σ as just a sum of the utilities of preferences satisfied in the agreement. Let s_i and s_j be a score representing the degree of preference satisfaction, then the global utility will be:

$$u_\beta = \sum s_i * u(\beta_i), u_\sigma = \sum s_j * u(\sigma_j) \quad (3)$$

Matchmaking Steps. Now we can outline the steps of the matchmaking process:

⁷ Without loss of generality, for the sake of simplicity in this example we consider supplies where only *hard constraints* have been set.

⁸ It is not in the scope of this paper to investigate on how to compute $u(\beta_1)$ and $u(\beta_2)$; we might assume, without loss of generality, they are determined in advance by means of either direct assignment methods (Ordering, Simple Assessing or Ratio Comparison) or pairwise comparison methods (like AHP and Geometric Mean) [20].

1: Every time a seller enters the marketplace, he proposes his supply expressing both *hard* and *soft constraints* (preferences). Eventually, for each preference σ_j (if any) he expresses the corresponding utilities $u(\sigma_j)$.

2: Similarly the buyer who enters the marketplace will express *hard* and *soft constraints* as well as the utility $u(\beta_i)$.

3: Based on buyer's and seller's specifications, the matchmaker returns a ranked list of agreements such that: [a] they satisfy both the *hard constraints* in the request and conversely their *hard constraints* are satisfied by the request; [b] the rank is evaluated taking into account preferences and utility functions u_β and u_σ as defined by equations (3).

In a P2P e-marketplace, the aim is to maximize both buyer's and seller's utilities in the final agreement, so the matchmaker has to propose agreements mutually beneficial for both of them. Such agreements are computed considering the higher values of u_β and u_σ **utilities product** [18].

4 Matchmaking with Vague Knowledge Bases

E-Marketplaces are typical systems where the notion of fuzziness is often involved. It is usual to find, among others, concepts like *Cheap* or *Expensive*. Similarly, numerical variables involved in a commercial transaction expose a fuzzy behavior. For instance, suppose to have a buyer looking for a car provided with a warranty greater than 100,000 kilometers and a supplier selling his car with a 80,000 kilometers warranty. If the buyer's warranty specification is modeled as preference, we can not say they do not match at all. Instead we can say they match with a certain degree.

Also notice that in both cases — conceptual and numerical information — the fuzziness is: (1) strongly dependent from the user point of view. The idea of "cheapness" changes moving from a user to another one; (2) allowed only in *soft constraints*. If the user expresses the willingness of selling a car with 80,000 kilometers warranty within *hard constraints*, it means that he does not want to negotiate on it at all.

A logical language able to allow the user to express also her fuzzy *soft constraints* would be then a good choice to model matchmaking. Given a DLR-Lite ontology \mathcal{O} and a set of facts \mathcal{F} we model a vague knowledge base $\mathcal{K}_{match} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$ for matchmaking in e-marketplaces where:

- \mathcal{P} represents user's (both buyer and seller) *soft constraints*. In \mathcal{P} 's rules, also roles and concepts from \mathcal{O} can be used.

In this case, \mathcal{F} represents the tuples of the relational database storing P2P advertisements. Notice that since we model a P2P marketplace, then advertisements can be either supplies or demands, depending on the "searcher" point of view. The former are used in case a buyer enters the marketplace the latter in case a seller decides to find promising requests.

To represent requester requirements in a vague knowledge base setting, we model *hard constraints* as a conceptual query over \mathcal{O} and *soft constraints* as a query over a vague Datalog program \mathcal{P} .

Hereafter we will use the following notation:

$$\begin{aligned} \text{hard constraints} &= \begin{cases} \beta(x, \mathbf{y}) , \text{ buyer's strict requirements} \\ \sigma(x, \mathbf{y}) , \text{ seller's strict requirements} \end{cases} \\ \text{soft constraints} &= \begin{cases} \beta_i(x, \mathbf{y}, s) \text{ or } \beta_i(x, s) , \text{ buyer's preferences} \\ \sigma_j(x, \mathbf{y}, s) \text{ or } \sigma_j(x, s) , \text{ seller's preferences} \end{cases} \end{aligned}$$

x is a single variable. It is usually instantiated with the key value of a database tuple; \mathbf{y} (if present) will be instantiated by numerical values. It is used whenever an agreement on numerical variables (price, km/years warranty, etc.) has to be reached; s is the score variable as defined in Section 2 [22]. It represents the score associated to fuzzy predicates involved in the body of the rules;

Notice that since a score is associated to each fuzzy predicate, we can compute the global utility based on the two utility functions in Section 3. Furthermore, the two queries σ and β model the minimal requirements the buyer and the seller want to be satisfied in order to accept the final agreement. Notice that, if seller and buyer set hard constraints in conflict with each other, the corresponding supply will not be retrieved. *Soft constraints* are modeled via Datalog predicates β_i for the buyer and σ_j for the seller, where each of them represents a sub-part of the buyer/seller preferences.

The use of x , \mathbf{y} and s should be clearer looking at how buyer's request in Example 2 is formalized:

$$\begin{aligned} \beta_A(x) &\leftarrow \text{PassengerCars}(x) \\ \beta_B(x) &\leftarrow \text{hasColor}(x, y), \text{Gray}(y) \\ \beta_B(x) &\leftarrow \text{hasColor}(x, y), \text{Black}(y) \\ \beta(x) &\leftarrow \beta_A(x), \beta_B(x) \\ \beta_1(x, p, s) &\leftarrow \text{hasPrice}(x, p), \\ &\quad LS(0, 100000, 14000, 16000, p, s) \\ \beta_2(x, p, kmw, s) &\leftarrow \text{KmWarranty}(x, kmw), \text{hasPrice}(x, p) \\ &\quad RS(0, 400000, 80000, 100000, kmw, s_1), \\ &\quad LS(0, 100000, 17000, 19000, p, s_2), \\ &\quad s = \max(1 - s_1, s_2) \end{aligned}$$

With respect to the previous encoding we notice that defining β_1 and β_2 here we can use two different *L-functions* to specify membership degrees for the variable p .

5 Top- k Retrieval for Matchmaking in Vague Knowledge Bases

Given a DLR-Lite ontology \mathcal{O} and a set of facts \mathcal{F} in a relational database, we can detail the matchmaking framework in a vague knowledge base $\mathcal{K}_{match} = \langle \mathcal{F}, \mathcal{O}, \mathcal{P} \rangle$. In order to formulate a query and rank all the retrieved results, what is still missing is how to put together both buyer's and seller's requirements. Then:

1. for each buyer's preference β_i , write the corresponding rule in vague Datalog where the head contains the predicate $\beta_i(x, \mathbf{y}, s)$ as shown in Section 4; add the rule to \mathcal{P} ; set the utility value $u(\beta_i)$ as shown in Section 3; The same is for the seller where the head of each rule is the predicate $\sigma_j(x, \mathbf{y}, s)$ and for each of them utility values are $u(\sigma_j)$

2. add to \mathcal{P} the rules:

$$\begin{aligned} Buyer(x, \mathbf{y}, u_\beta) &\leftarrow \beta_1(x, \mathbf{y}_1, s_1), \beta_2(x, \mathbf{y}_2, s_2), \dots, u_\beta = u(\beta_1) \cdot s_1 + u(\beta_2) \cdot s_2 + \dots \\ Seller(x, \mathbf{y}, u_\sigma) &\leftarrow \sigma_1(x, \mathbf{y}_1, s_1), \sigma_2(x, \mathbf{y}_2, s_2), \dots, u_\sigma = u(\sigma_1) \cdot s_1 + u(\sigma_2) \cdot s_2 + \dots \end{aligned}$$

where for each variable in \mathbf{y} in the head of one of the two previous rules, the same variable occurs in at least one of the arrays of the corresponding body: $\mathbf{y}_1, \mathbf{y}_2, \dots$;

3. encode buyer's *hard constraints* requirements as a conceptual query over \mathcal{O} . Rewrite the query as a conjunctive query where the query is denoted with $\beta(x, \mathbf{y}_\beta)$. The same is for the seller and his hard constraints $\sigma(x, \mathbf{y}_\sigma)$;
4. solve the **Top- k retrieval** problem:

$$ans_k(\mathcal{P}, Match) = \text{Top}_k\{\langle x, \mathbf{y}, u \rangle \mid \langle \mathbf{y}, u \rangle \in \text{Top}_1\{\langle x, \mathbf{y}', u' \rangle \mid \mathcal{P} \models Match(x, \mathbf{y}', u')\}\}.$$

where $Match$ is the conjunctive query

$$Match(x, \mathbf{y}, u) \leftarrow \beta(x, \mathbf{y}_\beta), Buyer(x, \overline{\mathbf{y}}_\beta, u_\beta), \sigma(x, \mathbf{y}_\sigma), Seller(x, \overline{\mathbf{y}}_\sigma, u_\sigma), u = u_\beta * u_\sigma$$

and for each variable in the array \mathbf{y} , the same variable occurs in $\mathbf{y}_\beta, \overline{\mathbf{y}}_\beta, \overline{\mathbf{y}}_\sigma$ or \mathbf{y}_σ .

Basically, for each key value x of the database, we compute the best match $\langle \mathbf{y}, u \rangle$ for it, *i.e.*, $\langle \mathbf{y}, u \rangle \in \text{Top}_1\{\langle x, \mathbf{y}', u' \rangle \mid \mathcal{P} \models Match(x, \mathbf{y}', u')\}$, and then rank the top- k key values.

Notice that the rank is computed considering the product of buyer's and seller's utilities as stated at the end of Section 3 in order to reach an agreement appealing both for the buyer and for the seller.

6 An Illustrative Example

Let us better clarify the approach with the aid of a tiny example. In Table 1 two possible offers stored in an e-marketplace database are presented. We call $CarTable$ the relation representing Table 1. Based both on these information and some specific do-

ID	MODEL	TYPE	PRICE	DISCOUNT	KM	COLOR	AIRBAG	INTERIOR TYPE	AIR COND	ENGINE FUEL
34	ALFA 156	Sedan	12000	20%	25000	Black	1	LeatherSeats	0	Diesel
1812	FORD FOCUS	StationVagon	13000	20%	20000	Gray	1	LeatherSeats	1	Gasoline

Table 1. The $CarTable$ relation with a sample set of offers.

main knowledge, we will write the corresponding vague knowledge base \mathcal{K}_{match} as explained in Section 4 and Section 5. In Figure 3 tuples related to the relation $CarTable$ are represented together with a DLR-Lite ontology \mathcal{O} extending the one presented in Example 1. Auxiliary rules in Figure 2 are introduced and used only for the sake of clarity and conciseness.

Now, suppose to have the following buyer's request:

$$\begin{aligned}
& \text{hasPrice}(x, p) \leftarrow \text{hasPossibleCarPrice}(x, p) \\
& \text{Kilometers}(x_1, x_6), \leftarrow \text{CarTable}(x_1, \dots, x_{11}) \\
& \text{CataloguePrice}(x_1, x_4), \leftarrow \text{CarTable}(x_1, \dots, x_{11}) \\
& \text{MinimalPrice}(x_1, y), \leftarrow \text{CarTable}(x_1, \dots, x_{11}), y = x_4 - x_4 \cdot x_5 \\
& \text{LS}(k_1, k_2, a, b, p, 1) \leftarrow k_1 \leq p \leq a \\
& \text{LS}(k_1, k_2, a, b, p, 0) \leftarrow b \leq p \leq k_2 \\
& \text{LS}(k_1, k_2, a, b, p, s) \leftarrow a < p < b, s = (b - p)/(b - a) \\
& \text{RS}(k_1, k_2, a, b, p, 0) \leftarrow k_1 \leq p \leq a \\
& \text{RS}(k_1, k_2, a, b, p, 1) \leftarrow b \leq p \leq k_2 \\
& \text{RS}(k_1, k_2, a, b, p, s) \leftarrow a < p < b, s = (p - a)/(b - a)
\end{aligned}$$

Fig. 2. Auxiliary rules

$$\begin{aligned}
& \text{CarTable}(34, \text{ALFA 156}, 2002, 12,000, 20\%, 25,000, \text{Black}, 1, \text{LeatherSeats}, 0, \text{Diesel}) \\
& \text{CarTable}(1812, \text{FORD FOCUS}, 2001, 13,000, 20\%, 20000, \text{Gray}, 1, \text{LeatherSeats}, 1, \text{Gasoline}) \\
& \text{hasPossiblePrice}(34, p), p \in \{12000, 11900, 11800, \dots, 9700, 9600\} \\
& \text{hasPossiblePrice}(1812, p), p \in \{13000, 12900, 12800, \dots, 10500, 10400\} \\
& \exists 1 : \text{CarTable} \sqsubseteq \text{Cars} \quad (\text{disjoint Mazda, AlfaRomeo, Ford}) \\
& \text{Sedan} \sqcup \text{StationWagon} \sqsubseteq \text{Cars} \quad (\text{disjoint Sedan, StationWagon}) \\
& \exists 9 : \text{CarTable} \sqsubseteq \text{Seats} \quad (\text{disjoint AirConditioning, NoAirConditioning}) \\
& \text{Mazda} \sqcup \text{AlfaRomeo} \sqcup \text{Ford} \sqsubseteq \text{CarMake} \quad (\text{disjoint LeatherSeats, VelvetSeats}) \\
& \text{LeatherSeats} \sqcup \text{VelvetSeats} \sqsubseteq \text{Seats} \quad \dots
\end{aligned}$$

Fig. 3. A part of the vague knowledge base \mathcal{K}_{match} used in the example.

[Hard Constraint] (β) I want a *sedan* or a *station wagon*.

[Soft Constraint] (β_1) I would like *air conditioning* if the car has *leather seats*. (β_2) Preferably I would like to pay *less than 11000 €*. (β_3) The car should have *less than 15000 km*.

[Preferences Utilities] $u(\beta_1) = 0.05$; $u(\beta_2) = 0.5$; $u(\beta_3) = 0.45$

Then we add to the vague Datalog program \mathcal{P} in \mathcal{K}_{match} the rules:

$$\begin{aligned}
& \beta_A(x) \leftarrow \text{Sedan}(x) \\
& \beta_A(x) \leftarrow \text{StationWagon}(x) \\
& \beta(x) \leftarrow \beta_A(x) \\
& \beta_1(x, 1) \leftarrow \text{NoLeatherSeats}(x) \\
& \beta_1(x, 1) \leftarrow \text{LeatherSeats}(x), \text{AirConditioning}(x) \\
& \beta_1(x, 0) \leftarrow \text{LeatherSeats}(x), \text{NoAirConditioning}(x) \\
& \beta_2(x, p, s) \leftarrow \text{hasPrice}(x, p), \\
& \quad \text{LS}(0, 100000, 11000, 13000, p, s) \\
& \beta_3(x, s) \leftarrow \text{Kilometers}(x, k), \\
& \quad \text{LS}(0, 400000, 15000, 20000, k, s) \\
& \text{Buyer}(x, p, u_\beta) \leftarrow \beta_1(x, s_1), \beta_2(x, p, s_2), \beta_3(x, s_3), \\
& \quad u_\beta = 0.05 \cdot s_1 + 0.5 \cdot s_2 + 0.45 \cdot s_3
\end{aligned}$$

Since we are in a P2P e-marketplace, also the seller can express hard and soft constraints. Looking at the information modeled in Table 1 we see that a soft constraint is expressed on price: the seller prefers to sell the car at the catalogue price, furthermore he may apply a discount. In this case also a constraint on price is set; in fact, he does not want to go down such defined discount (hard constraint). Without loss of generality, for the sake of clarity in this example we consider the same hard and soft constraints for all the sellers within the e-marketplace. Seller's requirements are then encoded in \mathcal{K}_{match} as:

$$\sigma(x, p) \leftarrow \text{hasPrice}(x, p), \text{CataloguePrice}(x, \text{catP}), \\ \text{MinimalPrice}(x, \text{minP}), \text{minP} \leq p \leq \text{catP}$$

$$\sigma_1(x, p, s) \leftarrow \text{hasPrice}(x, p), \text{CataloguePrice}(x, \text{catP}), \\ \text{MinimalPrice}(x, \text{minP}), \text{minP} \leq p \leq \text{catP}, \\ \text{RS}(0, 100000, \text{minP}, \text{catP}, p, s)$$

$$\text{Seller}(x, p, u_\sigma) \leftarrow \sigma(x, p), \sigma_1(x, p, s_1), u_\sigma = s_1$$

Notice that, in this particular case, the specification of $u(\sigma_1)$ is not necessary because of equation (3).

According to the encoding in Section 5 the query is:

$$\text{Match}(x, p, u) \leftarrow \beta(x), \text{Buyer}(x, p, u_\beta), \sigma(x, p), \text{Seller}(x, p, u_\sigma), u = u_\beta \cdot u_\sigma$$

Solving $\text{ans}_2(\mathcal{K}_{match}, \text{Match})$ retrieval problem with respect to \mathcal{K}_{match} defined in this example, the ranked list of agreements is:

x	p	u
34	11300	0.3010
1812	11800	0.1885

Then, the agreement between ALFA 156 and the request is ranked better than FORD FOCUS.

7 Related Work

Recently, the problem of matchmaking has been investigated under different perspectives and many approaches have been proposed. An initial approach to matchmaking can be dated back to vague query answering [17] where the need to go beyond pure relational databases was addressed using weights attributed to several search variables. More recently similar approaches have been proposed extending SQL with "preference" clauses, in order to allow relaxed queries in structured databases [11] where only buyer's preferences are taken into account while retrieving promising supplies: no agreement is proposed as a result of the query process. In our framework we model the matchmaking process in a P2P marketplace, taking into account not only the buyer's preferences, but also the seller's ones, finding the most promising agreements w.r.t. preferences of them both. Classified-ads matchmaking, at a syntactic level, was proposed in [21] and [25] to perform a matchmaking between semi-structured descriptions. Approaches to matchmaking using LOOM as description language can be found, among others, in [2]

and [9]. Due to the growing interest in the Semantic Web initiative many approaches to matchmaking have been proposed in the framework of DAML+OIL, OWL and their grounding logical languages in particular Description Logics (DL). Matchmaking as satisfiability of concept conjunction in DLs was first proposed in [10]. In the framework of Retsina Multiagent infrastructure [24], a specific language was defined for agent advertisement, and matchmaking engine was developed [19], which carries out the process on five possible match levels. The approach in [19] was later extended in [15], where two new levels for matching classification were introduced. A similar classification was proposed — in the same venue — in [8], along with properties that a matchmaker should have in a DL based framework, and algorithms to classify and semantically rank matches within classes. An initial DL-based approach, adopting penalty functions ranking, has been proposed in [4], in the framework of dating systems. An extended matchmaking approach, with negotiable and strict constraints in a DL framework has been proposed in [7], using both concept contraction and concept abduction. The need to work in some way with approximation and ranking in DL-based approaches to matchmaking has also recently led to adopting fuzzy-DLs, as in SMART [1] or hybrid approaches, as in the OWLS-MX matchmaker [13]. SMART is a semantic matchmaking portal, based on fuzzy-DLs, able to deal with approximation in the requests description handled by crisp DL-reasoners. Nevertheless in such approaches the matchmaking process is defined according to buyer's perspective. In [16] a language able to express conditional preferences is proposed to perform a matchmaking in Description Logics. Also in this case nothing is said on how to compute an agreement — as needed in P2P scenarios. Furthermore, the notion of fuzzy/vague requirements is not addressed.

8 Conclusion

In this work we propose a semantic matchmaking approach that mixes various knowledge representation technologies to find the most promising agreements in a P2P e-marketplace. In particular, by exploiting ontologies in DLR-Lite and fuzzy rules we are able to model both *hard constraints* and *soft constraints*, while taking into account both buyer's and seller's preferences and utilities to find matches mutually beneficial for them both. The information needed for the P2P matchmaking process are modeled as a vague knowledge base, taking into account domain knowledge, while keeping the approach effective and scalable. A prototype is currently being implemented to further validate the approach through large scale experiments.

References

1. S. Agarwal and S. Lamparter. smart - a semantic matchmaking portal for electronic markets. In *Proc. of 7th Int. IEEE Conference on E-Commerce Technology*, 2005.
2. Y. Arens, C. A. Knoblock, and W. Shen. Query Reformulation for Dynamic Information Integration. 6:99–130, 1996.
3. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

4. A. Cali, D. Calvanese, S. Colucci, T. Di Noia, and F. M. Donini. A description logic based approach for matching user profiles. In *Proc. of DL'04*, volume 104 of *CEUR Workshop Proceedings*, 2004.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of AAAI'05*, 2005.
6. S. Ceri, G. Gottlob, and L. Tanca. *Logic Programming and Databases*. Springer Verlag, 1990.
7. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Concept Abduction and Contraction for Semantic-based Discovery of Matches and Negotiation Spaces in an E-Marketplace. *Electronic Commerce Research and Applications*, 4(4):345–361, 2005.
8. T. Di Noia, E. Di Sciascio, F. M. Donini, and M. Mongiello. A system for principled match-making in an electronic marketplace. *International Journal of Electronic Commerce*, 8(4):9–37, 2004.
9. Y. Gil and S. Ramachandran. PHOSPHORUS: a Task based Agent Matchmaker. In *Proc. International Conference on Autonomous Agents '01*, pages 110–111. ACM, 2001.
10. J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proc. of ADL-2001*, volume 44. CEUR Workshop Proceedings, 2001.
11. B. Hafenrichter and W. Kießling. Optimization of relational preference queries. In *In Proc. of ADC'05*, pages 175–184, Newcastle, Australia, Jan. 2005.
12. R. L. Keeney and H. Raiffa. Decisions with multiple objectives - preferences and value trade-offs. *Cambridge University Press*, 1993.
13. M. Klusch, B. Fries, M. Khalid, and K. Sycara. Owls-mx: Hybrid owl-s service matchmaking. In *Proc. of 1st Int. AAAI Fall Symposium on Agents and the Semantic Web*, 2005.
14. C. Li, K. C.-C. Chang, I. F. Ilyas, and S. Song. RankSQL: query algebra and optimization for relational top-k queries. In *Proc. of SIGMOD-05*, pages 131–142, New York, NY, USA, 2005. ACM Press.
15. L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. of WWW '03*, 2003.
16. T. Lukasiewicz and J. Schellhase. Variable-strength conditional preferences for matchmaking in description logics. In *Proc. of KR 2006*, 2006.
17. A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Trans. Office Inf. Syst.*, 6(3):187–214, 1988.
18. J. F. Nash. The bargaining problem. *Econometrica*, 18 (2):155–162, 1950.
19. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *Proc. of ISWC'02*. 2002.
20. J. Pomerol and S. Barba-Romero. *Multicriterion Decision Making in Management*. Kluwer Series in Operation Research. Kluwer Academic, 2000.
21. R. Raman, M. Livny, and M. Solomon. Matchmaking: distributed resource management for high throughput computing. In *Proc. of IEEE High Performance Distributed Computing Conf.*, 1998.
22. U. Straccia. Towards top-k query answering in deductive databases. In *Proc. of SMC-06*, pages 4873–4879. IEEE, 2006.
23. U. Straccia. Towards top-k query answering in description logics: the case of DL-Lite. In *Proc. of JELIA-06*, 2006.
24. K. Sycara, M. Paolucci, M. Van Velsen, and J. Giampapa. The RETSINA MAS infrastructure. *Autonomous agents and multi-agent systems*, 7:29–48, 2003.
25. D. Veit, J. Muller, M. Schneider, and B. Fiehn. Matchmaking for Autonomous Agents in Electronic Marketplaces. In *Proc. International Conference on Autonomous Agents '01*, pages 65–66. ACM, 2001.
26. L. A. Zadeh. Fuzzy sets. *Information and Control*, 8(3):338–353, 1965.