A New Approach To Semantic Matchmaking Using Description Logics

TOUFIK TAIBI

Freelance Software Developer
753 Village Green Ave
London N6K1H3, Ontario, Canada
email:ttaibi@gmail.com

Abstract. An agent wishing to delegate some of its tasks to others needs to use the services of a matchmaking agent. The quality of the matchmaking process represents a key success factor in the effectiveness of the delegation process. As more services are exposing descriptions based on languages developed in the framework of Semantic Web, there is no doubt that Descriptions Logics (DLs) on which Web Ontology Language (OWL-DL) is based will be playing a major role in this context. This work contributes to the field in three ways. First, the DL $\mathcal{ALCN}$ was used in a matchmaking process that yields a justified ranking of results. Second, the matchmaking process provides useful information to the requester/provider in order to modify or refine the original request/capability description for a better result. Third, the requester agents should be able to further filter the results based on other criteria that we call "service goodness" that are based on some Quality of Service (QoS) criteria as well as attributes such as trust and cost.

Keywords: Semantic matchmaking, description logics, software agents, semantic web, delegation, quality of service.

(Received November 01, 2009 / Accepted September 15, 2010)

1 Introduction

The Semantic Web was introduced to extend the Web in such a way that its content can expressed in a format that can be understood by machines through the creation of domain ontologies. This has increased both openness and interoperability in the web environment.

The widespread availability of resources and services has led to the complexity of finding the best matches for a given request. The need for an effective matchmaking process has become a key factor in many e-marketplaces in which supply and demand have to be matched and services need to be discovered.

In this context, we define an agent as an entity that acts on behalf of a user or another program. Agents need to delegate some tasks to others either because they don't have the capabilities or resources to perform them or that they want to optimize the execution of their tasks. Before the delegation process happens a requester agent (or a broker acting on its behalf if a full-mediation service is used) needs to use the services of a resource discovery agent (matchmaker) which matches requests with available offers. The purpose of the matchmaking process is to find, for a given request, the best matches available among the list of offers. The Semantic Web paradigm requires that descriptions of requests and offers should be in a structured form based on ontologies. We assume in what follows that the descriptions of requests and offers pertain to a common ontology.

Although matchmaking has been widely studied in the past, there has been recently a growing effort aimed at formalizing the process using Description Logics (DLs) [4]. DLs can model structured descriptions of requests and offers as concepts. Moreover, DLs relies

on the *open-world assumption* in which the absence of information is distinguished from negative information.

Usually, DL-based approaches exploit standard reasoning services of a DL system (satisfiability and subsumption) to match requests with offers. If an offer (supply) is described by a concept Sup and a request (demand) is described by a concept Dem, unsatisfiability is denoted by the empty conjunction of Sup and Dem, while satisfiability is denoted by its non-emptiness. Hence, unsatisfiability identifies incompatibility between requests and offers, while satisfiability identifies a potential match between them (compatibility). Dem subsumes Sup (every individual of concept Sup is also of concept Dem) means that requirements on Dem are completely fulfilled by Sup.

The matchmaking process needs to go beyond compatible and incompatible matches by providing an explained ranking of the most prominent matches. Moreover, when an exact match is not found (which is the case most of the time), the matchmaking process should identify what needs to be changed in Sup and/or Dem in order to get a better result. The automation of such process is still a widely accepted challenge.

This work goes beyond providing a list of ranked matches by allowing the requester to further filter the results based on other criteria that we call "service goodness" that are based on some Quality of Service (QoS) criteria as well as attributes such as trust, and cost that are embedded into a common capability description language called Requester Provider Capability Description Language ($\mathcal{RPCDL}$).

The rest of the paper is organized as follows. Section 2 provides an overview of DLs, while section 3 presents how matchmaking is performed using DLs. Section 4 defines the extra filtering process performed by a requester after receiving a ranked list of offers. Section 5 presents a case study applying our approach to matchmaking. Section 6 presents related work, while section 7 concludes the paper.

2 Description Logics

DLs are a family of logic formalisms for Knowledge Representation [7]. This section is intented to present the core principles of DLs. In DLs, the basic syntax elements are:

- Concepts, such as `Person`, `Female`, `Parent`, `Women` and `Mother`.
- Roles, such as `has_child`.
- Individuals, such as `Elizabeth` and `Charles`.

Intuitively, concepts represent sets of individuals and roles link individuals in different concepts, as the role `has_child` links a `Parent` with a `Person`. Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. In this paper we use Attributive Language with Complements and unquantified Number restrictions $\mathcal{ALCN}$. Although having more constructors makes a DL more expressive, this comes with a cost which is the explosion in computational complexity of inference services [6]. Hence a trade-off is necessary. $\mathcal{ALCN}$ meets the trade-off because it is both sufficiently expressive and yet offers PSpace-complete complexity of inference services [20]. Moreover, $\mathcal{ALCN}$ can be mapped into a subset of OWL-DL [2]. The following lists the constructs of $\mathcal{ALCN}$:

- $\top$ *universal concept* represents all individuals in the domain.
- $\bot$ *bottom concept* represents the empty set.
- A *atomic concept* represents all individuals belonging to the set represented by A.
- $\neg C$ *negation* represents all individuals not belonging to the set represented by C.
- $C \sqcap D$ *intersection* represents the individuals belonging to both to C and D.
- $C \sqcup D$ *union* represents the individuals belonging to either C or D.
- $\forall R.C$ *universal restriction* represents all individuals participating in relation R whose range are all individuals belonging to C.
- $\exists R.C$ *existential restriction* represents some individuals participating in relation R whose range are all individuals belonging to C.
- $(\geq\ nR)$,$(\leq\ nR)$ *unquantified number restriction* represents the minimum and the maximum individuals participating in relation R. We write $(=\ nR)$ for $(\geq nR) \sqcap (\leq nR)$.

Concept expressions can be used in *axioms* which can be either *containment* (symbol: $\sqsubseteq$), or *definition* (symbol: $\equiv$). Axioms impose restrictions on possible interpretations according to the knowledge elicited for a given domain. Definition can be expressed using containment as follows: $C \equiv D$ can be defined by: $C \sqsubseteq D$, $D \sqsubseteq C$.

A DL knowledge base typically comprises two components, a "TBox" and an "ABox". The TBox contains terminological knowledge (hence the term "TBox") and is built through declarations (axioms) that describe general properties of concepts. The ABox contains assertional knowledge (hence the term ABox)– knowledge that is specific to the individuals of a domain. Terminological knowledge is usually thought not to change (timeless), while assertional knowledge is usually thought to be subject to occasional or even constant change. For example, a TBox could include the following axioms: `Parent` $\equiv$ `Person` $\sqcap\exists$ `has_child.Person` and `Women` $\sqsubseteq$ `Person`, while an ABox could include the following assertions: `mother(Elizabeth)`, `person(Charles)` and `has_child(Elizabeth,Charles)`.

Let's turn our attention now to the semantics of $\mathcal{ALCN}$. A model provides a set of domain elements and a way to interpret each piece of syntax. For example, we can interpret `mother(Elizabeth)` and `has_child(Elizabeth,Charles)` only when we are told who `Elizabeth` is and who `Charles` is and what `Mother` means (a set of individuals) and what `has_child` means (a role, a binary relation). Whether or not the statements are true depends on whether `Elizabeth` is a `Mother` and on whether `Elizabeth`'s child is `Charles` in our domain. In model theory, formulas are fixed and interpretations are varied (as to tackle different domains). When a set of formulas is true in some domain, the formulas are said to represent a model of that domain. They say something accurate about it, but don't tell us everything. What makes this useful is that when we do some syntactic manipulation to generate new formulas from the model, we expect that the new thing we found is also true in the domain.

DLs semantic can be defined by the standard Tarski-style interpretations. 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 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 $a \neq b$. Table 1 and Table 2 define both the syntax and semantics of $\mathcal{ALCN}$ constructs and its TBox axioms respectively. A model of a TBox $\mathcal{T}$ is an interpretation satisfying all axioms of $\mathcal{T}$.

Table 1: Syntax and Semantics of $\mathcal{ALCN}$ constructs

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
Universal quantification	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
Existential quantification	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Number restrictions	$(\geq nR)$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\} \geq n\}$
	$(\leq nR)$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}}\} \leq n\}$

Table 2: Syntax and Semantics of $\mathcal{ALCN}$ TBox assertions

Name	Syntax	Semantics
Definition	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$

DL-based systems usually provide two basic reasoning services [17]:

- **Concept Satisfiability**: given a TBox $\mathcal{T}$ and a concept C, does there exist at least one model of $\mathcal{T}$ assigning a non-empty extension to C? Satisfiability is defined as follows: $C \not\sqsubseteq_{\mathcal{T}} \bot$.
- **Subsumption**: given a TBox $\mathcal{T}$ and two concepts C and D, is $C^{\mathcal{I}}$ always contained in $D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$? Subsumption between C and D is defined as follows: $C \sqsubseteq_{\mathcal{T}} D$. Subsumption can be expressed using concept unsatisfiability as follows: $C \sqcap \neg D \sqsubseteq_{\mathcal{T}} \bot$.

Intuitively, unsatisfiability means that a concept does not belong to a given ontology, whereas D subsumes C (or C is subsumed by D) ,means that D is more general than C and C is more specific than D. This means that C inherits all properties of D in addition to having its own properties. For example if in a certain domain the concepts `Person` and `Parent` are made of the following individuals: `Person`={`Elizabeth, Charles, William`}, and `Parent`={`Elizabeth,Charles`}. The individuals `Elizabeth` and `Charles` are in the set `Parent` because they participate in role `has_child`, while `William` does not.

3 Semantic Matchmaking

In the context of this work, matchmaking is defined as the process by which a resource discovery agent provides a justified ranking of advertisements matching

a certain request from a requester. Requests and advertisements are expressed with reference to a shared ontology (having TBox $\mathcal{T}$). A study of the latest advances in the field reveals the following classification of matches [17]:

- **Exact match**: $Sup \equiv_{\mathcal{T}} Dem$.This means that Sup and Dem are equivalent.
- **Full match**: $Sup \sqsubseteq_{\mathcal{T}} Dem$. This means that Dem subsumes Sup. Sup has at least all features required by Dem.
- **Plug-in match**: $Dem \sqsubseteq_{\mathcal{T}} Sup$. This means that Sup subsumes Dem. Dem may have features not fulfilled by Sup.
- **Intersection match**: $Dem \sqcap Sup \not\sqsubseteq_{\mathcal{T}} \bot$. This means that Sup and Dem have something in common and no conflicting features.
- **Disjoint match**: $Dem \sqcap Sup \sqsubseteq_{\mathcal{T}} \bot$. This means that there is a conflict between Dem and Sup.

The above degrees of matches are organized in an ascending order of preference. We now introduce the process of matchmaking a request with advertisements. A DL reasoner [1] is used to compute a hierarchy of all advertised services. For an incoming request R, a DL reasoner is used to classify R i.e. compute R's subsumption relationships with all advertised services. Advertisements equivalent to R are considered to be "exact matches", those subsumed by, but not equivalent to R are considered to be "full matches", those subsuming but not equivalent to R are considered to be "plug-in" matches. The DL reasoner is then used to classify $\neg R$. Advertisements subsuming but not equivalent to $\neg R$ are considered to be "intersection matches", while those subsumed by $\neg R$ are considered to be "disjoint matches".

Getting a ranked list of advertisements matching its request is not enough for a requester. The requester (provider) should be given extra-information as to allow it to refine/modify its request (advertisement). In order to do so, we need first to normalize the way in which concepts are defined. Every satisfiable concept C can be divided into three components: $C \equiv_{\mathcal{T}} C_c \sqcap C_r \sqcap C_n$. $C_c \equiv_{\mathcal{T}} A_1 \sqcap \ldots \sqcap A_h$ represents the conjunction of h atomic concepts. $C_r \equiv_{\mathcal{T}} QR_l.C_l \sqcap \ldots \sqcap QR_p.C_p$. Here $l < p$ and Q represents either $\exists$ or $\forall$. C_r represents the conjunction of $(p - l)$ concepts of the form $\forall R.D$ or $\exists R.D$, where D is a normalized concept. Finally, $C_n \equiv_{\mathcal{T}} (OR_s) \sqcap \ldots \sqcap (OR_t)$ represents the conjunction of all number restrictions. O is an operator that could be either $\geq$ or $\leq$. Each role can have at most two conjuncts of number restrictions of either $\geq$ or $\leq$.

Now, we are ready to define the refinement techniques for each of the of non-exact matches defined above.

- **Case of Full match**: Since $Sup \sqsubseteq_{\mathcal{T}} Dem$, then $Sup \equiv_{\mathcal{T}} Dem \sqcap C_l \sqcap \ldots \sqcap C_p$, where $l < p$. The matchmaker sends the part $C_l \sqcap \ldots \sqcap C_p$ to the requester in order to let it know the extra features provided by the provider of the service in order to put it into consideration (if required) in future requests.
- **Case of plug-in match**: Since $Dem \sqsubseteq_{\mathcal{T}} Sup$, then $Dem \equiv_{\mathcal{T}} Sup \sqcap C_l \sqcap \ldots \sqcap C_p$, where $l < p$. The matchmaker sends the part $C_l \sqcap \ldots \sqcap C_p$ to both requester and provider in order to improve the request or advertisement. The requester will know the requested features that are not supported by the provider of the service, while the provider can provide extra features to meet the demands of the requester. This leads to the following two cases.
  - request refinement: If we remove the part $C_l \sqcap \ldots \sqcap C_p$ from Dem we get an exact match ($Dem \equiv_{\mathcal{T}} Sup$).
  - advertisement refinement: We can use concept abduction [9] by finding a $\mathcal{ALCN}$ formula H satisfiable with respect to $\mathcal{T}$ such that $Sup \sqcap H \sqsubseteq_{\mathcal{T}} Dem$.
- **Case of intersection match**: Since, $Dem \sqcap Sup \not\sqsubseteq_{\mathcal{T}} \bot$, then $Dem \equiv_{\mathcal{T}} C \sqcap C_{l_1} \sqcap \ldots \sqcap C_{p_1}$ and $Sup \equiv_{\mathcal{T}} C \sqcap C_{l_2} \sqcap \ldots \sqcap C_{p_2}$. $l_1 < p_1$ and $l_2 < p_2$. C represents, the "common" features between Sup and Dem. There are two alternatives to consider in this case, each is based on whether the requester or provider takes the initiative of refining its request or advertisement.
  - request refinement: The requester keeps the common part (C) and remove the part which is different. We get, $Sup \sqsubseteq_{\mathcal{T}} C$, where C represents the refined Dem. So we get a full match.
  - advertisement refinement: The provider keeps the common part (C) and removes the part which is difference. We get $Dem \sqsubseteq_{\mathcal{T}} C$, where C represents the refined Sup. So we get a plug-in match, which should be treated as per the previous case.

- **Case of disjoint match**: In this case concept contraction [9] can be used to refine the request in oder to get an "intersection" match. We need to find two $\mathcal{ALCN}$ formulas G (for give-up) and K (for keep) such that $Dem \equiv_{\mathcal{T}} G \sqcap K$ and $K \sqcap Sup \not\sqsubseteq_{\mathcal{T}} \bot$.

4 Filtering Matchmaking Results Using "Service Goodness"Attributes

Our approach goes beyond providing a list of ranked matches by allowing the requester to further filter the results based on other criteria that we call "service goodness" that are based on some Quality of Service (QoS) criteria as well as attributes such as trust, and cost. To achieve this requests and advertisements should be written using a common capability description language named Requester Provider Capability Description Language
($\mathcal{RPCDL}$). Table 3 describes the high-level EBNF grammar of $\mathcal{RPCDL}$.

Table 3: EBNF Grammar of $\mathcal{RPCDL}$

$< Request\_Capability >::=< Context > < Description >$
$< Service\_Goodness >$
$< Context >::=< Domain\ of\ the\ ontology >$
$< Description >::=< An\ \mathcal{ALCN}\ formula\ defining$
$a\ request\ or\ capability >$
$< Service\_Goodness >::=< QoS > < Other\_Attributes >$
$< QoS >::=< Availability > < Accessibility > < Integrity >$
$< Performance > < Reliability >$
$< Other\_Attributes >::=< Trust > < Cost >$

As mentioned earlier requests and advertisements should use the same context (ontology describing the domain). If they have different contexts then an integration process need to be performed as defined for example in [21].

The filtering process starts at the requester immediately after it receives a ranked list of advertisements that match its request as described in the previous section. The "service goodness" attributes used are defined below:

- **Availability**: Availability reflects if the service is ready for immediate use. It represents the probability that a service is available. Mean Time To Repair ($MTTR$), usually associated with availability represents the mean time it takes to repair a service that has failed. Smaller values of $MTTR$ are desirable.

- **Accessibility**: Accessibility represents the degree in which a service is capable of serving a request. It represents the probability of service initiation at a point in time. There could be situations when a service is available but not accessible. High accessibility of services can be achieved by building highly scalable systems. Scalability refers to the ability to consistently serve the requests despite variations in the volume of requests. Again, accessibility can be calculated as:

$$\frac{the\ time\ the\ service\ was\ actually\ accessible}{time\ elapsed\ since\ the\ service\ started\ after\ being\ advertised}$$

- **Integrity**: Integrity represents how the service maintains the correctness of the interaction in respect to the requester. Proper execution of service transactions will provide the correctness of interaction. A transaction refers to a sequence of tasks that make-up a service which needs to be treated as a single unit of work. All the tasks have to be completed to make the transaction successful, otherwise all changes made should be rolled back. Integrity can be calculated as:

$$\frac{the\ number\ of\ transactions\ successfully\ completed}{total\ number\ of\ handled\ transactions}$$

  Here by "successfully" we mean that the provider has given a correct service result to the requester.

- **Performance**: Performance is defined as made-up of two attributes: throughput and latency. Throughput represents the number of service requests served per unit of time. Latency is the round-trip time between sending a request and receiving the response. Higher throughput and lower latency values represent good performance of a service.

- **Reliability**: Reliability represents the degree of being capable of maintaining a service and its associated quality. The Mean-Time Between Failures ($MTBF$) represents a measure of reliability of a service.

- **Trust**: Trust can be defined as the degree of confidence that an entity is capable of acting reliably and securely in a particular transaction. Trust management thus involves the collection of information necessary for defining trust values of entities and continuously monitoring and adjusting such values. Reputation mechanisms represent an attractive way of handling trust [22]. In these mechanisms, a requester will have an aggregated trust value (between 0 and 1) for each provider.

- **Cost**: This represent the amount of money the provider is charging for providing the service.

The filtering technique uses a weighting strategy in which the requester assigns a weighting factor ($\in [0, 1]$) reflecting the importance each of the eight attribute has for the requester. Weights are chosen such that

$\sum_{i=1}^{7} W_i = 1$. The requester also defines the relative difference between its requested value of each attribute and the value offered by the provider. If A_d is the value requested by a requester and A_s is a value offered by a provider then the percentage in difference ($PD \in [0,1]$)(from the viewpoint of the requested) is calculated as $PD = \frac{A_d - A_s}{A_d}$. A negative value of PD means that what is offered is more than what is expected and a positive value means the opposite. A final value of preferences (called $Pref \in [0,1]$) is calculated as follows:

$$Pref = \sum_{i=1}^{8} W_i * PD_i$$

$Pref$ combines weights and percentages in difference of all eight attributes. The final result is compared with a threshold value that the requester has. If $Pref \leq threshold$ then the requester will consider using the service of the provider, otherwise it will move to calculate $Pref$ for the next ranked provider.

The final result of the entire filtering process should be a set of providers instead of only one to cater for the dynamism of the system. If only one provider is generated, there is a possibility that by the time the requester wants to call its related service, the provider has either unadvertised for its service or has left the system.

5 Case Study

In this section, we use a modified version of the sample ontology that appeared in [17] to illustrate our matchmaking process. The ontology is related to the buying/selling of computers and their variants. Table 4 represents a TBox of the ontology.

Table 4: TBox of Computers Ontology

Monitor ≡ CRTmonitor ⊔ LCDmonitor
CRTmonitor ⊓ LCDmonitor ≡ ⊥
StorageDevice ≡ HardDisk ⊔ FlashDisk
OperatingSystem ≡ Linux ⊔ Windows
Device ≡ Monitor ⊔ StorageDevice
Software ≡ OperatingSystem ⊔ Browser ⊔ WordProcessor
Computer ≡ PC ⊔ Laptop
Computer ⊑ (≥ 1 hasStorageDevice)
⊓ ∀hasStorageDevice.StorageDevice
⊓ ∀hasSoftware.Software ⊓ (≥ 1 hasRAM)
HomePC ⊑ PC ⊓ (≥ 1 hasSoftware) ⊓ (= 1 hasOS)
⊓ (≥ 1 hasMonitor) ⊓ ∀hasMonitor.Monitor
Server ⊑ Computer ⊓ (≥ 2 hasCPU) ⊓ ∀hasRAM.(≥ 512 hasMB)
⊓ ∀hasStorageDevice.(≥ 20000 hasMB)

Based on the above TBox, let's take now examples of each of the four cases of matches.

- **Case of full match**: $Sup \equiv$ HomePC ⊓ ∀hasMonitor.LCDMonitor and $Dem \equiv$ HomePC. This is a full match because $Sup \sqsubseteq Dem$. In such a case the matchmaker send the part ∀hasMonitor.LCDMonitor to the requester in order to let it know that all provided HomePCs come with an LCD monitor. In such a case it is up to it to accept this or not.

- **Case of plug-in match**: $Sup \equiv$ HomePC ⊓ ∀hasMonitor.LCDMonitor and $Dem \equiv$ HomePC ⊓ ∀hasMonitor.LCDMonitor ⊓ ∀hasRAM.(≥ 1024 hasMB). This is plug-in match because $Dem \sqsubseteq Sup$. If the requester removes the part ∀hasMonitor.LCDMonitor ⊓ ∀hasRAM.(≥ 1024 hasMB) from Dem then we get full-match as $Sup \sqsubseteq Dem$. If the provider adds ∀hasRAM.(≥ 1024 hasMB) to Sup then we get an exact match (although we can add a more specific conjunct and we get a full match). However, in most cases it is easier for a requester to remove features from its request than for a provider to add features because sometimes it does not have the capability to offer them.

- **Case of intersection match**:$Sup \equiv$ HomePC ⊓ ∀hasRAM.(≥ 1024 hasMB ⊓ ∃hasSoftware.WordProcessor) and $Dem \equiv$ HomePC ⊓ ∀hasMonitor.CRTMonitor. HomePC $\equiv Dem \sqcap Sup$. If the requester keeps HomePC and removes the rest then $Sup \sqsubseteq Dem$ and we have a full match. If the provider keeps HomePC and removes the rest we get a plug-in match which can be refined as per the previous point.

- **Case of disjoint match**: $Sup \equiv$ HomePC ⊓ ∀hasMonitor.LCDMonitor and $Dem \equiv$ Server ⊓ ∀hasMonitor.CRTMonitor. Clearly $Dem \sqcap Sup \equiv \bot$. In this case we need to use concept contraction by finding two $\mathcal{ALCN}$ formulas G (for give-up) and K (for keep) such that $Dem \equiv G \sqcap K$ and $K \sqcap Sup \not\sqsubseteq_{\mathcal{T}} \bot$. In our case $K \equiv$ HomePC and $G \equiv$ ∀hasMonitor.LCDMonitor.

Now assuming that we have a ranked list of providers that can somehow fulfill the request of the requester, the next step is to further filter them based on "service goodness". In Table 5, we list the weights and PD of each of the eight attributes as defined by a requester. If the threshold of the requester is 0.1 the final value of $Pref = 0.02$ will be acceptable.

Table 5: Sample Filtering Process

W_i	PD_i	$W_i * PD_i$
$W_1 = 0.1$	$PD_1 = -0.25$	-0.025
$W_2 = 0.1$	$PD_2 = 0.3$	0.03
$W_3 = 0.1$	$PD_3 = 0.25$	0.025
$W_4 = 0.1$	$PD_4 = 0.25$	0.025
$W_5 = 0.1$	$PD_5 = -0.25$	-0.025
$W_6 = 0.1$	$PD_6 = -0.3$	-0.03
$W_7 = 0.20$	$PD_7 = 0.35$	0.07
$W_8 = 0.20$	$PD_8 = -0.2$	-0.04

$$Pref = \sum_{i=1}^{8} W_i * PD_i = 0.03$$

6 Related Work

With the growing information overload on the Web, matchmaking has been increasingly investigated in recent years under a number of perspectives and for different purposes. In this section, we will summarize work done on matchmaking in general (for the sake of completeness), with a focus on DL-based approaches. However, at the end of the section we compare our work with only those using DLs for matchmaking.

In [11],KQML was proposed as an agent communication language that can facilitate the matchmaking process. In [15], two developed matchmakers were described, the SHADE matchmaker, which operates over logic-based and structured text languages, and the COINS matchmaker, which operates over free text. These matchmakers have been used for a variety of applications, most significantly, in the domains of engineering and electronic commerce. Similar methods were later re-considered in the GRAPPA system [26]. Classified ads matchmaking, at a syntactic level, was proposed in [19] to matchmake semi-structured descriptions advertising computational resources in a fashion anticipating grid resources brokering. Matchmaking was used in SIMS [3] to dynamically integrate queries using KQML, and LOOM as description languages. LOOM was also used in the subsumption matching addressed in [12]. InfoSleuth [14], a system for discovery and integration of information, included an agent matchmaker, which adopted KIF as a description language and the deductive database language LDL++.

Matchmaking using satisfiability in DLs was first proposed in [13], [10] and [25]. In [23], the authors introduced a specific language (LARKS) for agent advertisement in the framework of the RETSINA Multi-agent infrastructure. Moreover, a matchmaking engine was developed which carries out the process using five filters. Such levels exploit both classical text-retrieval techniques and semantic match using θ-subsumption. However, standard features of a semantic-based system, such as satisfiability check were not used. Nonetheless, the authors introduced the notion of plug-in match in order to overcome the limitations of exact match. This was further extended in [18]. In [16], the authors extended the approach defined in [23] by adding two new levels of matching classification (subsumes and intersection). In [5], the authors proposed an approach to semantic matchmaking that uses concept difference [24], followed by a covering operation optimized using hypergraph techniques, in the framework of web services discovery. An initial DL-based approach, adopting penalty functions ranking, has been proposed by [8] in the framework of dating systems. An extended matchmaking approach based on concept contraction and concept abduction was presented in [9] and further extended in [17].

Our work can be seen as an extension to [16] and [9]. However, besides the similarities, there are major differences in the approach taken. Both similarities and differences can be summarized as follows:

- Our approach has similar classification of matchmaking levels found in [16], although different naming conventions were used. It is worth noting that [16] did not tackle request refinements. Our approach, uses techniques to refine a request which is somehow similar to what is defined in [9]. However, there are many differences between the two approaches (see below).

- The work in [9] uses concept contraction to improve the match from "disjoint" to "intersection". It also uses concept abduction to improve the match from "intersection" to "full". In contrast, our approach tackles all cases (besides exact match) and proposes ways to refine the match. In the case of "full" match the requester is informed of the "extra" features of the provided service so that it can use them in the future. "Plug-in","intersection" and "disjoint" matches can be improved to "full" matches by either making changes to the request or the advertisement.

- After the requester gets the matchmaking results, we have devised a filtering process based on "service goodness" which combines QoS attributes with other important attributes such as trust and cost. Those attributes are embedded in matchmaker reply as part of an ($\mathcal{RPCDL}$) message.

7 Conclusion

We have addressed the matchmaking problem between requests and advertisements from a DL perspective. We cannot overemphasize the importance of efficient and effective matchmaking in the Semantic Web.

The problem of matchmaking has been tackled from different aspects. First, we organized the degrees of matches by order of importance then defined ways to improve the request or advertisement in order to reach either an exact or full match. Then, we went beyond the matchmaking done at resource discovery level and defined a filtering process of the results based on attributes that we call "goodness of service". These attributes combine both QoS attributes and other attributes such as trust, and cost. To this end, we defined $\mathcal{RPCDL}$ as a language in which requests and advertisements can be written and "goodness of service" attributes can be conveyed by the providers. The outcome of the filtering process can trim-down tremendously the list of matching advertisements and provides the most accurate matches that the requester needs. To our knowledge, our refinement mechanism and the filtering using "service goodness" represent new contribution to the matchmaking field.

References

[1] Dl reasoners. available from http://www.cs.man.ac.uk/ sattler/reasoners.html.

[2] Owl. available from http://www.w3.org/tr/owl-features/.

[3] Arens, Y., Knoblock, C. A., and Shen, W. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems*, 6:99–13, 1996.

[4] Baader, F., Calvanese, D., D.and Mc Guinness, Nardi, D., and Patel-Schneider, P., editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[5] Benatallah, B., Hacid, M., Rey, C., and Toumani, F. Request rewriting-based web service discovery. In *Proceedings of International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 2003.

[6] Brachman, R. and Levesque, H. The tractability of subsumption in frame-based description languages. In *Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84)*, pages 34–37. Morgan Kaufmann, Los Altos, 1984.

[7] Brachman, R. and Levesque, H. *Knowledge Representation and Reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[8] Cali, A., Calvanese, D., Colucci, S., Di Noia, T., and Donini, F. M. A description logic based approach for matching user profiles. In *In Proceedings of the 17th International Workshop on Description Logics (DL04)*, volume 104, 2004.

[9] Colucci, S., Di Noia, T., Di Sciascio, E., Donini, F., and Mongiello, M. 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.

[10] Di Sciascio, E., Donini, F., Mongiello, M., and Piscitelli, G. A knowledge-based system for person-to-person e-commerce. In *In Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44, 2001.

[11] Finin, T., Fritzson, R., McKay, D., and McEntire, R. Kqml as an agent communication language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM94)*, pages 456–463. ACM Press, 1994.

[12] Gil, Y. and Ramachandran, S. Phosphorus: a task based agent matchmaker. In *Proceedings of International Conference on Autonomous Agents 01*, pages 110–111. ACM Press, 2001.

[13] Gonzales-Castillo, J., Trastour, D., and Bartolini, C. Description logics for matchmaking of services. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44, 2001.

[14] Jacobs, N. and Shea, R. Carnot and infosleuth database technology and the web. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 443–444. ACM Press, 1995.

[15] Kuokka, D. and Harada, L. Integrating information via matchmaking. *Journal of Intelligent Information Systems*, 6:261–279, 1996.

[16] Li, L. and Horrocks, I. A software framework for matchmaking based on semantic web technology. In *In Proc. International World Wide Web Conference (WWW 03)*, pages 331–339, 2003.

[17] Noia, T. D., Sciascio, E. D., and Donini, F. M. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.

[18] Paolucci, M., Kawamura, T., Payne, T., and Sycara, K. Semantic matching of web services capabilities. In *proceedings The Semantic Web - ISWC 2002*, volume 2342 of *Lecture Notes in Computer Science*, pages 333–347. Springer-Verlag, 2002.

[19] Raman, R., Livny, M., and Solomon, M. Matchmaking: distributed resource management for high throughput computing. In *Proceedings of IEEE High Performance Distributed Computing Conference*, pages 140–146, 1998.

[20] Schmidt-Schaub, M. and Smolka, G. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.

[21] Shvaiko, P. and Euzenat, J. A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171, 2005.

[22] Singh, A. and Liu, L. Trustme: Anonymous management of trust relationships in decentralized p2p systems. In *Proceedings of IEEE Peer-to-Peer Computing (P2P) 2003*, pages 142–149, 2003.

[23] Sycara, K., Widoff, S., Klusch, M., and Lu, J. Larks: Dynamic matchmaking among heterogeneus software agents in cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.

[24] Teege, G. Making the difference: A subtraction operation for description logics. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR94)*, pages 540–550.

[25] Trastour, D., Bartolini, C., and Priest, C. Semantic web support for the business-to-business e-commerce lifecycle. In *Proceedings of International World Wide Web Conference (WWW) 02*, pages 89–98, 2002.

[26] Veit, D., Muller, J., Schneider, M., and Fiehn, B. Matchmaking for autonomous agents in electronic marketplaces. In *Proceedings of International Conference on Autonomous Agents 01*, pages 65–66. ACM Press, 2001.