

Extending Semantic-Based Matchmaking via Concept Abduction and Contraction

Tommaso Di Noia¹, Eugenio Di Sciascio¹, and Francesco M. Donini²

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

² Università della Tuscia, via San Carlo, 32, I-01100, Viterbo, Italy
donini@unitus.it

Abstract. Motivated by the need to extend features of semantic matchmaking between request and offer descriptions, a model is presented that exploits recently proposed non-standard inference services in Description Logics.

The model allows to manage negotiable and strict constraints of a request (equivalently of an offer) while performing a matchmaking process, even if both the request and the offer are incompatible –some part of one description is in conflict with the other– and some constraints in one description are not specified in the other one.

An algorithm is presented to compute both which part of the request should be retracted and which part of the offer has to be refined in order to make them completely satisfiable with each other.

1 Introduction

The problem of matching request and offer descriptions arises in several scenarios. Among them web-services discovery, e-marketplaces, personnel recruitment and job assignment, dating agencies. All these scenarios share a common purpose: given a request, find among available descriptions those best fulfilling it, or “at worst”, when nothing better exists, those that fulfill at least some of the requirements.

Exact, or full, matches are usually rare and the true matchmaking process is aimed at providing one or more best available matches to be explored, in order to initiate a negotiation/transaction process. Non-exact matches should take into account both missing information – details that could be positively assessed in a second phase – and conflicting information – details that could leverage negotiation if the proposed match is worth enough pursuing. Obviously, when several matches are possible, a matchmaking service should rank them in a most-promising order, so as to maximize the probability of a successful match within the first trials.

Recently, motivated by the ongoing transformation of the “unstructured” Web in the Semantic, machine understandable one, Description Logics (DLs) have been investigated to model the matchmaking problem in various scenarios [39, 19, 35, 32, 33, 20, 6], which we discuss in the final Section. DLs formalization has several advantages; among them an open-world assumption can be made, incomplete information is admitted, and absence of information can be distinguished from negative information. In all proposals on matchmaking exploiting DLs formalization, constraints about an offer or a request are expressed as concepts O and R in a chosen DL. Then, a DL reasoner is used to

check (a) satisfiability of the conjunction $O \sqcap R$ — which corresponds to compatibility between the constraints of O and R — and (b) subsumption $O \sqsubseteq R$ — which corresponds to O fulfilling all requirements of R , and vice versa for $R \sqsubseteq O$. However, when several offers O_1, \dots, O_n are compatible with a request R (or vice versa) a *ranking* of counteroffers should be provided, using some rational criterion, possibly based on the logical specification of O_1, \dots, O_n and R , and when unsatisfiability ensues a user might like to know which constraints in the request caused it.

Usually, DL approaches exclude the case when a request R is inconsistent with the concept describing an offer O , assuming that all requirements are strict ones. Nevertheless other approaches not based on DLs [36] are much more liberal on this subject, allowing a user to specify negotiable requirements — some of which could be bargained in favor of others. In practice, there can be cases when a request is expressed by a user as a description where some of the requirements are strict ones, while other might be more loose and negotiable.

In [18, 15] Concept Abduction and Concept Contraction have been proposed as non-standard inference services in DL, to capture in a logical way the reasons why a counteroffer O_1 should be ranked better than a counteroffer O_2 for a given request R , and vice versa. In a nutshell, when a request is issued, Concept Contraction captures the possibility to relax some constraints of R when they are in conflict with an offer O — that is, when $O \sqcap R$ is an unsatisfiable concept. Intuitively, relaxable constraints are negotiable preferences, that could be dropped if the offer O satisfies some more important requirements of R . On the other hand, Concept Abduction captures the reasoning mechanism — namely, making hypotheses — involved when some constraints required by R are not specified in O — that obviously in later stages of the request/offer interaction might turn out to be fulfilled or not.

Here we propose a model that exploits Concept Contraction and Abduction and is able to take into account also incompatible pairs O, R , which are usually discarded by a matchmaking facilitator, thus easing discovery of negotiation spaces. Based on the model an algorithm is devised, which computes both the part of the request that should be retracted and the part of the offer that should be refined in order to make them completely satisfiable with respect to other.

The remaining of the paper is so structured. Section 2 revises Description Logics basics and the logic adopted here. Then Section 3 describes the non-standard inference services for DL we use. In Section 4 our logical setting is motivated and presented. Then we show in Section 5 how Concept Abduction and Contraction can be exploited to deal with negotiable and strict requirements in an extended matchmaking scenario. A discussions on relevant related work closes the paper.

2 Description Logics

DLs are a family of logic formalisms for Knowledge Representation [9, 22, 3] whose basic syntax elements are

- *concept* names, e.g., computer, CPU, device, software,
- *role* names, like hasSoftware, hasDevice
- *individuals*, like HPworkstationXW, IBMThinkPad, CompaqPresario.

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 is identified by the operators set it is endowed with. 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. Expressive DLs [14] are built on the simple \mathcal{AL} (Attributive Language) adding constructs in order to represent more expressive concepts.

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 union 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. 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 \mathcal{T} is an interpretation satisfying all inclusions and definitions of \mathcal{T} .

Adding new constructors increases DL languages expressiveness. Nevertheless, it is a well known result [11] that this usually leads to an explosion in computational complexity of inference services. Hence a trade-off is necessary. In this paper we refer to an \mathcal{ALN} (Attributive Language with unqualified Number restrictions) Description Logic and to *simple-TBox* modeled as set of axioms in which the left side can be only a concept name (both for inclusion and definition). Although limited, such a subset already allows a user to specify negotiable and strict constraints, to verify their consistency, and to hypothesize the feasibility of a negotiation process for a given offer, as we show in the following sections. In Table 1 we present the constructs of \mathcal{ALN} . Note that ontologies are usually designed as *simple-TBox* in order to express the relations among objects in the domain. Ontologies using the above logic can be easily modeled using languages for the Semantic Web. The strong relations between DLs and the above introduced languages for the Semantic Web [4] is clearly present in the definition of the OWL language, particularly in its sub-language OWL-DL where expressiveness is allowed, while trying to keep computational completeness and decidability. The subset of OWL-DL TAGs allowing to express an \mathcal{ALN} DL is presented in Table 3. In the rest of the paper we will use DL syntax instead of OWL-DL syntax, because the former is more compact. Nevertheless all the examples and the simple ontology we use to model them, can be rewritten using OWL DL syntax.

Table 1. Syntax and semantics of the constructs of \mathcal{ALN}

name	syntax	semantics
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \setminus A^{\mathcal{I}}$
universal quantification	$\forall R.C$	$\{d_1 \mid \forall d_2 : (d_1, d_2) \in R^{\mathcal{I}} \rightarrow d_2 \in C^{\mathcal{I}}\}$
number restrictions	$(\geq n R)$	$\{d_1 \mid \#\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geq n\}$
	$(\leq n R)$	$\{d_1 \mid \#\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leq n\}$

Table 2. Syntax and semantics of the TBox assertions

name	syntax	semantics
definition	$A = C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$
inclusion	$A \sqsubseteq C$	$A^{\mathcal{I}} \subseteq C^{\mathcal{I}}$

Table 3. Correspondence between OWL and DL syntax

OWL syntax	DL syntax
$\langle owl : Thing / \rangle$	\top
$\langle owl : Nothing / \rangle$	\perp
$\langle owl : Classrdf : ID = "C" / \rangle$	C
$\langle owl : ObjectPropertyrdf : ID = "R" / \rangle$	R
$\langle rdfs : subclassOf / \rangle$	\sqsubseteq
$\langle owl : equivalentClass / \rangle$	\equiv
$\langle owl : disjointWith / \rangle$	\sqsupset
$\langle owl : intersectionOf / \rangle$	\sqcap
$\langle owl : allValuesFrom / \rangle$	\forall
$\langle owl : maxCardinality / \rangle$	\leq
$\langle owl : minCardinality / \rangle$	\geq

3 Non-standard Inferences for \mathcal{ALN} DL

All DL systems provide subsumption and satisfiability as standard reasoning services.

1. *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 ?
2. *Subsumption*: given a TBox \mathcal{T} and two concepts C and D , is C more general than D in any model of \mathcal{T} ?

Although this two basic services are very useful in several scenarios, there are cases where there is the need to overcome subsumption and satisfiability.

In our matchmaking scenario –where obviously an open world assumption is made – if constraints in \mathbf{R} are not specified in \mathbf{O} , and then subsumption does not hold, it should be possible to hypothesize missed ones and ask to refine \mathbf{O} giving information on them. On the other hand we may not want to discard an offer \mathbf{O} with respect to a request \mathbf{R} on the basis of the conjunction inconsistency, *i.e.*, where they present mutually conflicting features. A user may like to know both why they are in conflict and which are the features that are not compatible, in order to establish, in case, a negotiation process based on such characteristics.

The logical formalization of negotiable requirements can be based on Concept Contraction [15] — Contraction has been formalized by Gärdenfors’ [25] as the first step in belief revision — and Concept Abduction [18].

In the following we highlight basic properties of these non-standard DL inferences, and we refer to [17] for a thorough presentation, in the framework of a tableaux-based approach.

Starting with the concepts \mathbf{O} and \mathbf{R} , if their conjunction $\mathbf{O} \sqcap \mathbf{R}$ is unsatisfiable in the TBox \mathcal{T} representing the ontology, our aim is to retract requirements in \mathbf{R} , G (for *Give up*), to obtain a concept K (for *Keep*) such that $K \sqcap \mathbf{O}$ is satisfiable in \mathcal{T} .

Definition 1. *Let \mathcal{L} be a DL, \mathbf{O} , \mathbf{R} , be two concepts in \mathcal{L} , and \mathcal{T} be a set of axioms in \mathcal{L} , where both \mathbf{O} and \mathbf{R} are satisfiable in \mathcal{T} . A Concept Contraction Problem (CCP), identified by $\langle \mathcal{L}, \mathbf{R}, \mathbf{O}, \mathcal{T} \rangle$, is finding a pair of concepts $\langle G, K \rangle \in \mathcal{L} \times \mathcal{L}$ such that $\mathcal{T} \models \mathbf{R} \equiv G \sqcap K$, and $K \sqcap \mathbf{O}$ is satisfiable in \mathcal{T} . We call K a contraction of \mathbf{R} according to \mathbf{O} and \mathcal{T} .*

We use \mathcal{Q} as a symbol for a CCP, and we denote with $SOLCCP(\mathcal{Q})$ the set of all solutions to a CCP \mathcal{Q} . We note that there is always the trivial solution $\langle G, K \rangle = \langle \mathbf{R}, \top \rangle$ to a CCP. This solution corresponds to the most drastic contraction, that gives up everything of \mathbf{R} . In our matchmaking framework, it models the (infrequent) situation in which, in front of some very appealing offer \mathbf{O} , incompatible with the request, a user just gives up completely his/her specifications \mathbf{R} in order to meet \mathbf{O} .

On the other hand, when $\mathbf{O} \sqcap \mathbf{R}$ is satisfiable in \mathcal{T} , the “best” possible solution is $\langle \top, \mathbf{R} \rangle$, that is, give up nothing — if possible. Since usually one wants to give up as few things as possible, some minimality in the contraction must be defined. We do not delve into details, and just mention that there exists an algorithm, *i.e.*, $solveCCP(\mathbf{O}, \mathbf{R}, \mathcal{T})$ [17] to compute a minimal G *i.e.*, with respect to a minimality criterion, (and a maximal K) for a given offer \mathbf{O} with respect to a request \mathbf{R} and a TBox \mathcal{T} .

Once contraction has been applied, and consistency between the offer and the request has been regained, there is still the problem with partial specifications, that is, it could be the case that the offer — though compatible — does not imply the request. Then, it is necessary to assess what should be hypothesized in the offer in order to initiate a transaction with the requester. This non-standard inference is named *Concept Abduction*, in analogy to Charles Peirce’s Abduction [34, 31].

Definition 2. *Let \mathcal{L} be a DL, \mathbf{O} , \mathbf{R} , be two concepts in \mathcal{L} , and \mathcal{T} be a set of axioms in \mathcal{L} , where both \mathbf{O} and \mathbf{R} are satisfiable in \mathcal{T} . A Concept Abduction Problem (CAP), identified by $\langle \mathcal{L}, \mathbf{R}, \mathbf{O}, \mathcal{T} \rangle$, is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \models \mathbf{O} \sqcap H \sqsubseteq \mathbf{R}$, and*

moreover $O \sqcap H$ is satisfiable in \mathcal{T} . We call H a hypothesis about O according to D and \mathcal{T} .

Also for Concept Abduction, there exist algorithms [18, 17] *i.e.*, $solveCAP(O, R, \mathcal{T})$ that can compute H for \mathcal{ALN} concepts O, R and a simple TBox \mathcal{T} . A numerical version $rankPotential(O, R, \mathcal{T})$ of the algorithm also exists [20], which computes the length of a Concept Abduction H , thus providing a score to the similarity between an offer and a request. An evolution of the above approach is presented in [13], where penalty functions are introduced managing user interest levels on both R and O features.

We note that Concept Contraction extends satisfiability — in particular, by providing new concepts G and K when a conjunction $O \sqcap R$ is unsatisfiable — while Concept Abduction extends subsumption — in particular, by providing a new concept H when O is not subsumed by R .

4 DL Modeling of the Matchmaking Framework

Matchmaking systems are electronic intermediary systems that bring requests and offers together and usually provide various support services [23, 36, 37, 20]. Obviously, other techniques, not logic-based have been used to model such systems. Nevertheless, using standard database techniques to model a matchmaking framework, we would be obliged to completely align the attributes of the offer and request in order to evaluate a match. If requests and offers are simple names or strings, the only possible match would be identity, resulting in an all-or-nothing approach to matchmaking. Vector-based techniques based on classical Information Retrieval can be used, too, thus reverting matchmaking to similarity between weighted vectors of terms. Although effective for fixed technical domains, such approaches miss the fact that offers and requests usually have some sort of structure in them. Our logical approach allows users to state only part of the information about their offers, and moreover, to state information at different abstract levels, leaving to the logic apparatus the burden of comparing specified characteristics. The logic apparatus we consider is based on a reference ontology, *i.e.*, a set of axioms (TBox), modeling implicit knowledge on the domain and a matchmaking service endowed with the non-standard services described in the previous section. We stress that every argument in what follows could be restated exchanging request with offer, depending on the actor who is actively starting the search. In the examples that follow we will use information modeled using the simplified ontology proposed in Figure 1, which models descriptions for a tiny computer marketplace. In a generic matchmaking framework, a concept describing a request, can be read as a set of constraints of the user needs. A description modeling explicit knowledge on a request R represents a set of constraints on it. For example, the simple $R = \text{homePC} \sqcap (\geq 1 \text{ hasOS})$ represents the following constraints: $\{\text{homePC}, (\geq 1 \text{ hasOS})\}$.

The idea is that the active requesting/offering user (or agent), models the previous set as the conjunction of two different ones. The first set accounts for negotiable features, *i.e.*, what the requester may accept to retract to initiate a transaction. The second set represents non-negotiable elements, the quota of R the requester is most interested in and then does not want in conflict with O ; at least she/he may be willing to hypothesize some of them and, later in the process, request informations on the offer to the provider. In DL terms,

```

unix  $\sqsubseteq$   $\neg$ winX
linux  $\sqsubseteq$  unix
CRTmonitor  $\sqsubseteq$   $\neg$ LCDmonitor
USBpen  $\sqsubseteq$  removableDevice
AMD  $\sqsubseteq$   $\neg$ Intel
computer  $\sqsubseteq$  ( $\geq 1$  hasStorageDevice)  $\sqcap$  ( $\geq 1$  hasComponent)
server  $\equiv$  computer  $\sqcap$  ( $\geq 2$  hasCPU)
personalComputer  $\sqsubseteq$  computer
personalComputer  $\sqsubseteq$   $\neg$ PDA
homePC  $\equiv$  personalComputer  $\sqcap$  ( $\geq 1$  hasOS)  $\sqcap$  ( $\leq 1$  hasOS)  $\sqcap$  ( $\leq 1$  hasCPU)

```

Fig. 1. The toy ontology used as reference in examples

we model the non-negotiable and the negotiable constraints as a conjunction of concepts: negotiable constraints, from now on \mathcal{NG} and non-negotiable, strict, constraints, from now on \mathcal{ST} . Basically, negotiable constraints express a continued interest in an agreement with a counterpart also when no potential match is available and on the basis of a relaxation of the constraint [36]. Obviously, if an element belongs to \mathcal{NG} it cannot belong to \mathcal{ST} *i.e.*, if (≥ 1 hasOS) is a negotiable constraint, then it cannot be also a non-negotiable one, otherwise an inconsistency ensues within the user specification. Such an inconsistency may be caused by the interplay between the ontology and the user's specifications about negotiable/non-negotiable constraints. For example, considering the previous description $\mathcal{NG} = \text{homePC}$ and $\mathcal{ST} = (\geq 1 \text{ hasOS})$, due to the ontology axiom related to `homePC`, one obtains (≥ 1 hasOS) both in \mathcal{NG} and \mathcal{ST} . Clearly an incoherent specification of what is negotiable and what is not. Scenarios as the one presented previously, can be managed using Concept Abduction and considering both \mathcal{NG} and \mathcal{ST} as conjunctions of \mathcal{ALN} concepts. Solving the following **CAP**: $\mathcal{NG} \sqcap H_1 \sqsubseteq \mathcal{ST}$, if $H_1 = \mathcal{ST}$ then the whole \mathcal{ST} has to be hypothesized to obtain a subsumption relation, *i.e.*, nothing of both the explicit and implicit information in \mathcal{NG} is in \mathcal{ST} . In our scenario if $H_1 \neq \mathcal{ST}$ the requester may be asked to reformulate her/his model of negotiable and not-negotiable information. Solving the **CAP** $H_1 \sqcap H_2 \sqsubseteq \mathcal{ST}$, H_2 represents the shared part between \mathcal{ST} and \mathcal{NG} .

Notice that we model \mathcal{NG} and \mathcal{ST} after **R** has been formulated. Before the user composes **R**, nothing is known about \mathcal{NG} and \mathcal{ST} . Our approach hence does not model a request like “I am looking for a personal computer possibly suited for domestic use and with necessarily an OS installed”. The previous request already embeds negotiable and strict requirements in its specification. Its logical model can be built using a logic and an approach like the one proposed in [13].

Here, first we build the logical model of the request, *e.g.*, “I am looking for a personal computer suited for domestic use and with an OS installed” (`homePC` \sqcap (≥ 1 hasOS)), and then the active user specifies \mathcal{NG} and \mathcal{ST} . This remark is necessary to take into account the ontology concept hierarchy. In fact if the user is looking for $\dots \sqcap \forall \text{hasOS} . (\text{linux} \sqcap \text{unix})$, due to the ontology axioms, actually s/he is looking for $\dots \sqcap \forall \text{hasOS} . \text{linux}$. Then it is not correct to specify $\mathcal{ST} = \dots \sqcap \forall \text{hasOS} . \text{linux}$ and $\mathcal{NG} = \dots \sqcap \forall \text{hasOS} . \text{unix}$ or vice versa because of the subsumption relation between `linux` and `unix`.

5 Matchmaking Algorithm

In the following we show how, using both the notions of Concept Abduction and Contraction together with the logical formalization of negotiable constraints, the search of negotiation spaces within a matchmaking framework is feasible.

The basic assumption in this formalization is that an actor is willing to play an active role, *i.e.*, s/he may be willing to consider retracting on some of the constraints expressed in the initial description to leverage negotiation.

With reference to Figure 1, consider $R = \text{computer} \sqcap \forall \text{hasCPU.Intel} \sqcap (\geq 1 \text{hasCPU})$, with $\mathcal{NG} = \forall \text{hasCPU.Intel}$ and $ST = \text{computer} \sqcap (\geq 1 \text{hasCPU})$, and $O = \text{homePC} \sqcap \forall \text{hasCPU.AMD}$. It is possible to verify that $R \sqcap O$ is unsatisfiable, due to CPU specification. A solution of the **CCP** $\langle \mathcal{ALN}, O, R, T \rangle$ is the pair $\langle G, K \rangle$ where $G = \forall \text{hasCPU.Intel}$ and $K = \text{computer} \sqcap (\geq 1 \text{hasCPU})$.

Note that the previous one is not “the” solution. In fact another solution pair is, for example, $G = (\geq 1 \text{hasCPU})$ and $K = \text{computer} \sqcap \forall \text{hasCPU.Intel}$. But a similar solution is not compatible with negotiable specification: the requester is not willing to retract on $(\geq 1 \text{hasCPU})$.

To catch this possibility a minimality criterion for a **CCP** would be: “do not retract concepts belonging to ST ”, *i.e.*, do not add to G concepts C so that $ST \sqsubseteq C$. By definition $K \sqcap O$ is satisfiable, hence, K potentially matches O .

Let us point out that, having $R \sqcap O$ unsatisfiable, if $ST \sqcap O$ is satisfiable, the unsatisfiability arises because of \mathcal{NG} (or to the conjunction of elements ST and other elements in \mathcal{NG}). In spite of some conflicting constraints in O , there is at least a part of ST (that is the most important one from the user’s point of view), which can be potentially satisfied by O , hypothesizing concepts not expressed there.

On the other hand if ST and O are unsatisfiable, there is no way to continue the matchmaking process, unless a reformulation of the request R or at least the negotiable preferences \mathcal{NG} . In the latter case suggestions on which part of ST has to be transformed into a negotiable constraint, can be made solving a **CCP** $\langle \mathcal{L}, ST, O, T \rangle$. The solution $\langle G, K \rangle$ can be interpreted as the part that must be set as negotiable (G), and the one remaining strict (K).

Actually, the above scenario keeps its significance also if we flip over R and O , *i.e.*, if we have an offer expressed as ST and \mathcal{NG} by the active actor.

Notice that, if \mathcal{NG} and ST are expressed in the demand R , a change in their specifications [see lines 7–15 in the algorithm below] requires recomputing previous matches. On the other hand, if \mathcal{NG} and ST are expressed in the supplies O s, no recomputing is needed.

In a more formal way, we propose the following algorithm to cope with the extended matchmaking scenario. The algorithm executes calls to *solveCCP* and *solveCAP* and takes as inputs:

- R : description of the active actor (either a request or an offer), with $R = ST \sqcap \mathcal{NG}$
- O : conversely defined description (with respect to the above item, either an offer or a request)
- T : the ontology describing the marketplace domain

Algorithm *matchmaker*($\mathbf{O}, \mathbf{R}, \mathcal{T}$)

input \mathcal{ALN} concepts \mathbf{O}, \mathbf{R} , where $\mathbf{R} = \mathcal{ST} \sqcap \mathcal{NG}$

output $\langle G, K, H_K \rangle$

begin algorithm

```

1:  if ( $\mathbf{R} \sqcap \mathbf{O}$  is unsatisfiable)
2:    if ( $\mathcal{ST} \sqcap \mathbf{O}$  is satisfiable){
3:       $\langle G, K \rangle = \text{solveCCP}(\mathbf{O}, \mathbf{R}, \mathcal{T})$ ;
4:       $H_K = \text{solveCAP}(\mathbf{O}, K, \mathcal{T})$ ;
5:      return  $\langle G, K, H_K \rangle$ ;
6:    }
7:    else{
8:      Ask the active actor to change
      preferences on strict constraints ;
9:      if YES {
10:        $\langle G_{\mathcal{ST}}, K_{\mathcal{ST}} \rangle = \text{solveCCP}(\mathbf{O}, \mathcal{ST}, \mathcal{T})$ ;
11:        $\mathcal{ST}_{\text{new}} = K_{\mathcal{ST}}$ ;
12:        $\mathcal{NG}_{\text{new}} = \mathcal{NG} \sqcap G_{\mathcal{ST}}$ ;
13:        $\mathbf{R}_{\text{new}} = \mathcal{NG}_{\text{new}} \sqcap \mathcal{ST}_{\text{new}}$ ;
14:       return matchmaker( $\mathbf{O}, \mathbf{R}_{\text{new}}, \mathcal{T}$ );
15:     }
16:     else return  $\langle -, \perp, \perp \rangle$ ;
17:   }
18: else{
19:    $H_D = \text{solveCAP}(\mathbf{O}, \mathbf{R}, \mathcal{T})$ ;
20:   return  $\langle \top, \mathbf{R}, H_D \rangle$ ;
21: }

```

end algorithm

The computational complexity of the above algorithm strictly depends on those of *solveCCP* and *solveCAP* [15, 17].

The algorithm *matchmaker* determines parameters useful to compare the relevance of a match between a request and several offers, returning the triple $\langle G, K, H_K \rangle$ of \mathcal{ALN} concepts:

- G , represents the part to be retracted from \mathbf{R} in order to obtain a compatible match
- K , what, in \mathbf{R} , can be satisfied by \mathbf{O}
- H_K , the hypothesis to be formulated to make \mathbf{R} completely satisfied by \mathbf{O}

The triple $\langle -, \perp, \perp \rangle$ is returned when the elements to give up in the transaction belong to \mathcal{ST} , *e.g.*, there are constraints the user does not want to retract on. $\langle -, \perp, \perp \rangle$ is a level of “unrecoverable” mismatch.

We would like to point out that it may look sufficient, at a first glance, $G = \mathbf{R}$. Nevertheless this result corresponds to the situation where $\mathcal{ST} = \top$, then $\mathcal{ST} \sqcap \mathbf{O}$ is still satisfiable and a *give up* is possible on the whole \mathbf{R} . Notice that in row 3, *matchmaker* solves a *CCP* on \mathbf{O} and \mathbf{R} rather than on \mathbf{O} and \mathcal{NG} . The rationale is easily understandable considering the example at the beginning of this section where the source of inconsistency

is the conjunction of concepts both in \mathcal{ST} and \mathcal{NG} and the *give up* operation is on concepts subsuming \mathcal{NG} .

A global numerical parameter, evaluating how promising is a match, can be evaluated using *rankPotential* [20], the numerical version of *solveCAP*, which allows to compute the length of a Concept Abduction $|H|$. A function depending on $|G|$, $|K|$, $|H_K|$, the “length” of G , K and H_K , can hence evaluate a score for each match between a request and a set of offers.

5.1 A Simple Example

In order to better describe our approach we present hereafter a simple example scenario, which we build with reference to the ontology in Figure 1. Let us consider the following supply:

S. Single processor PC with at least Linux pre-installed. The offer includes an LCD monitor and a scanner.

- $S = \text{personalComputer} \sqcap \forall \text{hasComponent.}(\text{LCDmonitor} \sqcap \text{scanner})$
 $\sqcap (\leq 1 \text{ hasCPU}) \sqcap \forall \text{hasOS.linux} \sqcap (\geq 1 \text{ hasOS})$

and the request:

D. I'm looking for a PC for domestic use, with Unix, equipped with a CRT monitor and USB pen for data storing.

- $D = \text{homePC} \sqcap \forall \text{hasOS.unix} \sqcap \forall \text{hasComponent.CRTmonitor}$
 $\sqcap \forall \text{hasStorageDevice.USBpen}$

Within the above request let us consider:

I absolutely need a PC for domestic use with Unix.

- $\mathcal{ST} = \text{homePC} \sqcap \forall \text{hasOS.unix}$

I would appreciate the CRT monitor and the USB pen.

- $\mathcal{NG} = \forall \text{hasComponent.CRTmonitor} \sqcap \forall \text{hasStorageDevice.USBpen}$

S and D are not consistent with respect to the reference ontology, τ , due to the monitor specification.

The *matchmaker*(S, D, τ) returns:

$G = \forall \text{hasComponent.CRTmonitor}$

$K = \text{homePC} \sqcap \forall \text{hasOS.unix} \sqcap \forall \text{hasStorageDevice.USBpen}$

$H_K = (\leq 1 \text{ hasOS}) \sqcap \forall \text{hasStorageDevice.USBpen}$

6 Discussion

In this paper we proposed a model that, by exploiting Concept Contraction and Abduction allows to manage negotiable and strict constraints of a request (equivalently of an offer) while performing a matchmaking process, even if both the request and the offer are incompatible and some constraints in one description are not specified in the other one. An algorithm was presented able to compute both which part of the request should be retracted and which part of the offer should be refined to make a pair request - offer completely satisfiable.

In the remaining of this section we comment on similar approaches and discuss them. In [24] and [30] matchmaking was introduced, based on KQML, as an approach whereby potential producers / consumers could provide descriptions of their products/needs to be later unified by a matchmaker engine to identify potential matches. The proposed solutions to this challenging issue reverted to either a rule based approach using the Knowledge Interchange Format (KIF) [26] (the SHADE [30] prototype) or a free text comparison (the COINS [30] prototype). Approaches similar to the cited ones were deployed in SIMS [1], which used KQML and LOOM as description language and Info-Sleuth [29], which adopted KIF and the deductive database language LDL++. LOOM is also at the basis of the subsumption matching addressed in [27].

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 [37] and [33] the LARKS language is proposed, specifically designed for agent advertisement. The matching process is a mixture of classical IR analysis of text and semantic match via Θ -subsumption. Nevertheless, a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match.

First approaches based on subsumption services offered by DL reasoners were proposed in [21, 28, 39]. In [19, 20] properties that a matchmaker should have in a DL based framework, were described and motivated, and algorithms to classify and rank matches into classes were presented, *i.e.*, *Exact match*: all requested characteristics are available in the description examined; *Potential match*: some part of the request is not specified in the description examined; *Partial match*: some part of the request is in conflict with the description examined. The algorithms are modified versions of the structural subsumption algorithm originally proposed in [10] and compute a distance between each description w.r.t. a request in each class. Matchmaking of web-services described in DAML-S, providing a ranking of matches based on the DL-based approach of [19] is presented in [16]. An extension to the approach in [33] was proposed in [32] where two new levels for service profiles matching are introduced. Notice that there the *intersection satisfiable* level is introduced, whose definition is close to the one of *potential matching* proposed in [19], but no measure of similarity among intersection satisfiable concepts is given.

Semantic service discovery via matchmaking in the Bluetooth [8] framework has been investigated in [35]. Also here the issue of approximate matches, to be somehow ranked and proposed in the absence of exact matches, was discussed, but as in the previous papers no formal framework was given. Instead a logical formulation should allow to devise correct algorithms to classify and rank matches.

Matching in DLs has been widely treated in [5] although with no relation to match-making. 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 in [7, 6] web services matchmaking was tackled. An approach was proposed, based on the Difference operator in DLs [38], followed by a set covering operation optimized using hypergraph techniques. The adopted DL is \mathcal{L}_1 . Notice that performing a difference operation needs a subsumption relation between descriptions to be matched, which instead is not required to solve a Concept Contraction Problem. This strict condition may make Concept Difference hard to use in a matchmaking process, where descriptions overlap is usually a sufficient condition to start the process. Anyway, to the best of our knowledge there is no algorithm able to compute an exact Concept Difference in a DL endowed of the negation constructor. In [12] an algorithm is proposed for Difference on approximation of concepts.

At a first glance, also the Least Common Subsumer (*lcs*) [2] could be useful to model the problem of finding negotiation spaces in a matchmaking framework. In fact with *lcs*, given two concepts D and C , it is possible to compute the concept representing all the properties that D and C have in common. Nevertheless computing an *lcs* may lead to loss of information. For example having $\mathbf{O} = \neg A \sqcap B$ and $\mathbf{R} = A \sqcap C$, we obtain $lcs = \top$. There is no way to recover information of B in \mathbf{O} and of C in \mathbf{R} .

Acknowledgments

The authors acknowledge partial support of projects PON CNOSSO and MS3DI.

References

1. Y. Arens, C. A. Knoblock, and W. Shen. Query Reformulation for Dynamic Information Integration. *Journal of Intelligent Information Systems*, 6:99–130, 1996.
2. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI2003)*, pages 319–324, 2003.
3. F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.
4. F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.
5. F. Baader, R. Kusters, A. Borgida, and D. Mc Guinness. Matching in Description Logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
6. B. Benatallah, M.-S. Hacid, C. Rey, and F. Toumani. Request Rewriting-Based Web Service Discovery. In *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 242–257. Springer, 2003.
7. B. Benatallah, M.-S. Hacid, C. Rey, and F. Toumani. Semantic Reasoning for Web Services Discovery. In *Proc. of Workshop on E-Services and the Semantic Web at WWW 2003*, May 2003.
8. Bluetooth. <http://www.bluetooth.com>.

9. A. Borgida. Description Logics in Data Management. *IEEE Transactions on Knowledge and Data Engineering*, 7(5):671–682, 1995.
10. A. Borgida and P. F. Patel-Schneider. A Semantics and Complete Algorithm for Subsumption in the CLASSIC Description Logic. *Journal of Artificial Intelligence Research*, 1:277–308, 1994.
11. R. Brachman and H. Levesque. 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.
12. S. Brandt, R. Küsters, and A.-Y. Turhan. Approximation and difference in description logics. In *Proc. International Conference on Principles of Knowledge Representation and Reasoning (KR2002)*, pages 203–214. Morgan Kaufman, 2002.
13. A. Cali, D. Calvanese, S. Colucci, T. D. Noia, and F. Donini. A Logic-based Approach for Matching User Profiles. In *Proc. of KES'2004 Intl. Conf. on Knowledge-Based Intelligent Information and Engineering Systems*, 2004. To appear.
14. D. Calvanese and G. De Giacomo. Expressive description logics. In *The Description Logic Handbook: Theory, Implementation and Applications*, pages 178–218. 2003.
15. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Concept Abduction and Contraction in Description Logics. In *Proceedings of the 16th International Workshop on Description Logics (DL'03)*, volume 81 of *CEUR Workshop Proceedings*, September 2003.
16. S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Logic Based Approach to web services discovery and matchmaking. In *Proceedings of the E-Services Workshop at ICEC'03*, September 2003.
17. S. Colucci, T. D. Noia, E. D. Sciascio, F. Donini, and M. Mongiello. Uniform Tableaux-Based Approach to Concept Abduction and Contraction in ALN DL. In *Proceedings of the 17th International Workshop on Description Logics (DL'04)*, volume 104 of *CEUR Workshop Proceedings*, 2004.
18. T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 337–342, Acapulco, Mexico, August 9–15 2003. Morgan Kaufmann, Los Altos.
19. T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. Semantic matchmaking in a P-2-P electronic marketplace. In *Proc. Symposium on Applied Computing (SAC '03)*, pages 582–586. ACM, 2003.
20. T. Di Noia, E. Di Sciascio, F. Donini, and M. Mongiello. A system for principled Matchmaking in an electronic marketplace. In *Proc. International World Wide Web Conference (WWW '03)*, pages 321–330, Budapest, Hungary, May 20–24 2003. ACM, New York.
21. E. Di Sciascio, F. Donini, M. Mongiello, and G. Piscitelli. A Knowledge-Based System for Person-to-Person E-Commerce. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44 of *CEUR Workshop Proceedings*, 2001.
22. 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.
23. M. Dumas, B. Benatallah, N. Russell, and M. Spork. A Configurable Matchmaking Framework for Electronic Marketplaces. *Electronic Commerce Research and Applications*, 3(1):95–106, 2004.
24. T. Finin, R. Fritzson, D. McKay, and R. McEntire. KQML as an Agent Communication Language. In *Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94)*, pages 456–463. ACM, 1994.
25. P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. Bradford Books, MIT Press, Cambridge, MA, 1988.

26. M. R. Genesereth. Knowledge Interchange Format. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference*, pages 599–600, Cambridge, MA, 1991. Morgan Kaufmann, Los Altos.
27. Y. Gil and S. Ramachandran. PHOSPHORUS: a Task based Agent Matchmaker. In *Proc. International Conference on Autonomous Agents '01*, pages 110–111. ACM, 2001.
28. J. Gonzales-Castillo, D. Trastour, and C. Bartolini. Description Logics for Matchmaking of Services. In *Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001)*, volume 44. CEUR Workshop Proceedings, 2001.
29. N. Jacobs and R. Shea. Carnot and Infosleuth – Database Technology and the Web. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 443–444. ACM, 1995.
30. D. Kuokka and L. Harada. Integrating Information Via Matchmaking. *Journal of Intelligent Information Systems*, 6:261–279, 1996.
31. H. Levesque. A Knowledge-Level Account for Abduction. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89)*, pages 1061–1067. Morgan Kaufmann, Los Altos, 1989.
32. L. Li and I. Horrocks. A Software Framework for Matchmaking Based on Semantic Web Technology. In *Proc. International World Wide Web Conference (WWW '03)*, pages 331–339, Budapest, Hungary, May 20–24 2003. ACM, New York.
33. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic Matching of Web Services Capabilities. In *The Semantic Web - ISWC 2002*, number 2342 in Lecture Notes in Computer Science, pages 333–347. Springer-Verlag, 2002.
34. C. . Peirce. Abduction and induction. In *Philosophical Writings of Peirce*, chapter 11. J. Buchler, 1955.
35. S. Avancha, A. Joshi, and T. Finin. Enhanced Service Discovery in Bluetooth. *IEEE Computer*, pages 96–99, 2002.
36. 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.
37. K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.
38. G. Teege. Making the difference: A subtraction operation for description logics. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 540–550. MK, 1994.
39. D. Trastour, C. Bartolini, and C. Priest. Semantic Web Support for the Business-to-Business E-Commerce Lifecycle. In *Proc. International World Wide Web Conference (WWW) '02*, pages 89–98. ACM, 2002.