Available online at www.sciencedirect.com

SCIENCE DIRECT®

ELSEVIER

Electronic Commerce Research and Applications 4 (2005) 345–361

Electronic Commerce Research and Applications

www.elsevier.com/locate/ecra

Concept abduction and contraction for semantic-based discovery of matches and negotiation spaces in an e-marketplace

Simona Colucci [a], Tommaso Di Noia [a], Eugenio Di Sciascio [a,*], Francesco M. Donini [b], Marina Mongiello [a]

[a] *Dipto di Elettrotecnica ed Elettronica, SisInf Lab, Politecnico di Bari, Via Orabona 4, 70125 Bari, Italy*

[b] *Università della Tuscia, Viterbo, Italy*

Received 14 April 2005; received in revised form 2 June 2005; accepted 28 June 2005
Available online 22 July 2005

Abstract

In this paper, we present a Description Logic approach – fully compliant with the Semantic web vision and technologies – to extended matchmaking between demands and supplies in a semantic-enabled Electronic Marketplace, which allows the semantic-based treatment of negotiable and strict requirements in the demand/supply descriptions. To this aim, we exploit two novel non-standard Description Logic inference services, Concept Contraction – which extends satisfiability – and Concept Abduction – which extends subsumption. Based on these services, we devise algorithms, which allow to find negotiation spaces and to determine the quality of a possible match, also in the presence of a distinction between strictly required and optional elements. Both the algorithms and the semantic-based approach are novel, and enable a mechanism to boost logic-based discovery and negotiation stages within an e-marketplace. A set of simple experiments confirm the validity of the approach.
© 2005 Elsevier B.V. All rights reserved.

Keywords: E-commerce; Negotiable constraints; Matchmaking; Description logics; Concept abduction; Concept contraction; Semantics; Semantic web

1. Introduction

Matchmaking is, in a nutshell, the process of searching the space of possible matches between generic Request and Offer descriptions to find the most promising ones. Collecting and matching

* Corresponding author. Tel.: +39 080596 3641; fax: +39 080596 3410.
E-mail addresses: s.colucci@poliba.it (S. Colucci), t.dinoia@poliba.it (T. Di Noia), disciascio@poliba.it (E. Di Sciascio), donini@unitus.it (F.M. Donini), mongiello@poliba.it (M. Mongiello).

complementary needs in order to leverage mutually beneficial transactions is a task at the core of an electronic marketplace, especially in peer-to-peer scenario [30,54].

The purpose of a matchmaking facilitator is then, basically, filtering those supplies (or conversely demands, depending on the point of view), which may be worth pursuing based on a given demand (supply). Obviously, a negotiation process may then ensue, up to the actual transaction.

The observation that usually descriptions are endowed of a structure and exact match is rare, makes obvious the need for exploiting methods and techniques able to give some kind of score to matches and eventually rank them.

Several recent proposals try to formalize with Description Logics (DLs), the matchmaking of supplies and demands in an electronic marketplace (see, among others [27,57,55,24,42,26,23]). DLs, in fact, allow for an open-world assumption. Incomplete information is admitted and absence of information can be distinguished from negative information (we provide a little detail on DLs in Section 2). Furthermore such languages allow to model constraints of structured descriptions as concepts, which share a common ontology.

The need for a common, shared, ontology is usually the main objection toward logic-based approaches to matchmaking. Nevertheless, it should be considered that even when requests and offers are expressed in heterogeneous forms, integration techniques [43,14] can be employed to make heterogeneous descriptions comparable; once they are reformulated in a comparable way, one is still left with true matchmaking problems: (i) given a proposal, are there any compatible counteroffers? (ii) in the presence of several counteroffers, which, and why, are the most promising ones?

Most logic-based approaches tend to use the standard reasoning services of a DL system – subsumption and (un)satisfiability – to classify potential partners. In brief, if a supply is described by a concept S and a demand by a concept D, unsatisfiability of the conjunction of S and D, noted as $S \sqcap D$, identifies the incompatible proposals, satisfiability identifies potential partners – that still have to agree on underspecified constraints – and subsumption of S and D, noted as $S \sqsubseteq D$, means that requirements on D are completely fulfilled by S.

As a matter of fact the flat classification into compatible and incompatible matches can be of little help in the presence of, say, some hundred compatible proposals.

Usually, proposed approaches exclude the case in which the concept expressing a demand is inconsistent with the concept expressing a supply, assuming that all requirements are strict ones. However, proposals for matchmaking outside DLs (e.g., [53]) 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. Let us consider, for example, a simplified scenario in an e-marketplace. We have a demand expressed as: *I am looking for a computer such that it* must *be a PC, with Linux installed, and* I would prefer *a single processor one, including a CRT monitor and USB pen for data storing*. We also have an available supply expressed as: *Single processor personal computer equipped with an high level LCD monitor available*. Considering the above demand, it is clear that the demander will be interested in supplies where specified information are compatible, at least, on the *Linux* and *PC* strict constraints. On other constraints specified in the demand, the demander may be willing either to negotiate – if they are incompatible with the supply description – or ask (equivalently hypothesize) if they are not specified within the demand. With respect to the above supply, the *CRT monitor* constraint might be retracted because of its incompatibility with the LCD monitor one in the supply. Furthermore, since it is not specified in the offer, the demander may be willing to ask the supplier information about the Operating System and USB pen specification.

In order to model such reasonable, commonsense, behavior in a semantic-based framework, in this paper we exploit Concept Contraction [17] and Concept Abduction [23], two novel non-standard inference services for DLs, and present algorithms to discover negotiation spaces and to determine the quality of a possible match, also in the presence of a distinction between strictly required and optional elements. The remaining of the paper is structured as follows. Next section revises DLs basics and the logic we adopt here. Then we present non-standard inference services we use and motivate the rationale underlying them. In Section 4, our logical setting is motivated and presented. Then, in Section 5, we show how non-standard inferences can be exploited to deal with negotiable and strict requirements in an extended matchmaking scenario, and finally provide a measure of the match quality. We also present an illustrative example and experiments validating the approach. Section 6 discusses related work on the subject. Conclusions close the paper.

2. Description logics

2.1. Preliminary notions

DLs are a family of logic formalisms for Knowledge Representation [10,28,3]. To make the paper self-contained, we briefly present introductory notions of DLs.

In DLs, the basic syntax elements are

- *concept* names, e.g., `computer`, `CPU`, `device`, `software`,
- *role* names, e.g., `hasSoftware`, `hasDevice`
- *individuals*, e.g., `HPworkstationXW`, `IBM-ThinkPad`, `CompaqPresario`.

Intuitively, concepts stand for sets of objects, and roles link objects in different concepts, as the role `hasSoftware` which links computers to softwares. Individuals are used for special named elements belonging to concepts.

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

Basic elements can be combined using *constructors* to form concept and role *expressions*, and each DL has its distinguished set of constructors. Every DL allows one to form a *conjunction* of concepts, usually denoted as $\sqcap$; some DL include also disjunction $\sqcup$ and complement $\neg$ to close concept expressions under boolean operations.

Roles can be combined with concepts using *existential role quantification*, e.g., $\mathtt{computer} \sqcap \exists\mathtt{hasSoftware.wordProcessor}$ which describes the set of computers whose softwares include a word processor, and *universal role quantification*, e.g., $\mathtt{server} \sqcap \forall\mathtt{hasCPU.Intel}$, which describes servers with only Intel processors on board. Other constructs may involve counting, as number restrictions: $\mathtt{computer} \sqcap (\leqslant 1\,\mathtt{hasCPU})$ describes computers with just one CPU, and $\mathtt{computer} \sqcap (\geqslant 4\,\mathtt{hasCPU})$ describes computers equipped with at least four CPUs. Many other constructs can be defined, increasing the expressive power of the DL, up to *n*-ary relations [15].

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, $(\forall R.C)^{\mathcal{I}} = \{d_1 \in \Delta | \forall d_2 \in \Delta : (d_1, d_2) \in R^{\mathcal{I}} \to d_2 \in C^{\mathcal{I}}\}$.

Concept expressions can be used in *inclusion assertions*, and *definitions*, which impose restrictions on possible interpretations according to the knowledge elicited for a given domain. For example, we could impose that monitors can be divided into CTR and LCD using the two inclusions: $\mathtt{monitor} \sqsubseteq \mathtt{LCDMonitor} \sqcup \mathtt{CRTMonitor}$ and $\mathtt{CRTMonitor} \sqsubseteq \neg\mathtt{LCDMonitor}$. Or, that computers for a domestic use have only one operating system as $\mathtt{homePC} \sqsubseteq (\leqslant 1\,\mathtt{hasOS})$. Definitions are useful to give a meaningful name to particular combinations, as in $\mathtt{server} \equiv \mathtt{computer} \sqcap (\geqslant 2\,\mathtt{hasCPU})$. 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 $C \equiv D$ when $C^{\mathcal{I}} = D^{\mathcal{I}}$. A *model* of a TBox $\mathcal{T}$ is an interpretation satisfying all inclusions and definitions of $\mathcal{T}$. DL-based systems usually provide at least two basic 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 $\mathcal{T}$?

2.2. $\mathcal{ALN}$ description logic

As it is obvious, adding new constructors makes DL languages more expressive. Nevertheless, it is a well-known result [12] 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 the $\mathcal{ALN}$ DL, which can be mapped in a subset of OWL-DL [46]. Although limited, such a subset already allows a user to specify negotiable and non-negotiable constraints, to verify their consistency, and to hypothesize the feasibility of a trade for a given counteroffer, as we show in the following sections.

Here, we present only the constructs of $\mathcal{ALN}$ (**A**ttributive **L**anguage with unqualified **N**umber restrictions) DL (see Table 1):

- $\top$ *universal concept*. All the objects in the domain.
- $\bot$ *bottom concept*. The empty set.
- A *atomic concepts*. All the objects belonging to the set represented by A.
- $\neg A$ *atomic negation*. All the objects not belonging to the set represented by A.
- $C \sqcap D$ *intersection*. The objects belonging both to C and D.
- $\forall R.C$ *universal restriction*. All the objects participating in the R relation whose range are all the objects belonging to C.
- $\exists R$ *unqualified existential restriction*. There exists at least one object participating in the relation R.[1]
- $(\geqslant nR)$, $(\leqslant nR)$, $(= nR)$[2] *unqualified number restrictions*. Respectively the minimum, the maximum and the exact number of objects participating in the relation R.

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

Name	Syntax	Semantics
Top	$\top$	$\Delta^{\mathcal{I}}$
Bottom	$\bot$	$\emptyset$
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	$(\geqslant nR)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \geqslant n\}$
	$(\leqslant nR)$	$\{d_1 \mid \sharp\{d_2 \mid (d_1, d_2) \in R^{\mathcal{I}}\} \leqslant n\}$

Table 2
Syntax and semantics of the TBox assertions

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

Ontologies are usually designed as *simple-TBox* in order to express the relations among objects in the domain. With a *simple-TBox* the left side is represented by a concept name in all the axioms (for both inclusion and definition) (see Table 2).

(1) definition
$\texttt{server} \equiv \texttt{computer} \sqcap (\geqslant 2\texttt{hasCPU})$
(2) inclusion
$\texttt{computer} \sqsubseteq (\geqslant 1\texttt{hasStorageDevice})$

Ontologies using the above logic can be easily modeled using languages for the Semantic Web. The basic idea of the Semantic Web initiative is to structure information with the aid of markup languages, based on the XML language, such as

[1] Notice that $\exists R$ is equivalent to $(\geqslant 1R)$.
[2] We write $(= nR)$ for $(\geqslant nR) \sqcap (\leqslant nR)$.

RDF and RDFS [51], DAML + OIL and more recently OWL [21,45,46]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of domain ontologies, and aim at increasing openness and interoperability in the web environment. The strong relations between DLs and the above referenced languages for the Semantic Web [4] is also evident in the definition of the OWL language. In fact there are three different sub-languages for OWL:

OWL-Lite. It allows class hierarchy and simple constraints on relation between classes.

OWL-DL. Based on DLs theoretical studies, it allows a great expressiveness keeping computational completeness and decidability.

OWL-Full. Using such a language, there is a huge syntactic flexibility and expressiveness. This freedom is paid in terms of no computational guarantee.

The subset of *OWL-DL* TAGs allowing to express $\mathcal{ALN}$ is presented in Table 3.[3] As an example we present a translation of an inclusion axiom in DL to OWL.

$$\texttt{onSalePC} \sqsubseteq \texttt{homePC} \sqcap (\leqslant 1 \texttt{hasOS}) \sqcap \forall \texttt{hasOS.winX}$$

⟨*owl* : *Classrdf:ID* = "*onSalePC*"/⟩
  ⟨*rdfs* : *subClassOf*⟩
    ⟨*owl* : *intersectionOfrdf*: *parseType* = "*Collection*"⟩
      ⟨*owl* : *Classrdf:ID* = "*homePC*"/⟩
      ⟨*owl* : *Restriction*⟩
        ⟨*owl* : *onPropertyrdf:resource* = "*hasOS*"/⟩
        ⟨*owl* : *maxCardinality* *rdf:datatype* = "*&xsd;nonNegativeInteger*"⟩
          1
        ⟨/*owl* : *maxCardinality*⟩
        ⟨*owl* : *allValuesFrom* *rdf:resource* = "*#winX*"/⟩
      ⟨/*owl* : *Restriction*⟩
    ⟨/*owl* : *intersectionOf*⟩
  ⟨/*rdfs* : *subClassOf*⟩
⟨/*owl* : *Class*⟩

[3] Since in $\mathcal{ALN}$ only *unqualified existential restriction* is allowed, then the OWL ⟨*owl* : *someValuesFrom*/⟩ restriction on an *ObjectProperty* must be ⟨*owl* : *Thing*/⟩.

Table 3
Correspondence between OWL and DL syntax

OWL syntax	DL syntax
⟨*owl* : *Thing*/⟩	$\top$
⟨*owl* : *Nothing*/⟩	$\bot$
⟨*owl* : *Classrdf:ID* = "*C*"/⟩	C
⟨*owl* : *ObjectPropertyrdf:ID* = "*R*"/⟩	R
⟨*rdfs* : *subClassOf*/⟩	$\sqsubseteq$
⟨*owl* : *equivalentClass*/⟩	$\equiv$
⟨*owl* : *disjointWith*/⟩	$\neg$
⟨*owl* : *intersectionOf*/⟩	$\sqcap$
⟨*owl* : *allValuesFrom*/⟩	$\forall$
⟨*owl* : *someValuesFrom*/⟩	$\exists$
⟨*owl* : *maxCardinality*/⟩	$\leqslant$
⟨*owl* : *minCardinality*/⟩	$\geqslant$
⟨*owl* : *cardinality*/⟩	$=$

In the rest of the paper, we will use DL syntax instead of OWL-DL syntax, for compactness reasons. Nevertheless all the examples and the ontology we use to model them, can be rewritten using OWL-DL syntax.

3. Non-standard inferences for extended matchmaking

All DL systems provide subsumption and satisfiability as standard inference services. In some scenarios, such as our matchmaking setting, these services are not sufficient for solving inference problems, as they basically provide boolean (yes/no) answers with no further explanation. Approaches for solving non-standard inference problems are usually based on characterization of subsumption. In the following, we briefly present non-standard inference services that will be used in the formalization of our approach to extended matchmaking, which includes the possibility to determine negotiation spaces between supply and demand.

The logical formalization of negotiable requirements we propose is based on Concept Contraction [17] – Contraction has been formalized by Gärdenfors' [32] as the first step in belief revision – and Concept Abduction [23]. In the following, we highlight basic properties of these non-standard inferences, and we refer to [19] for a thorough presentation, in the framework of a tableaux-based approach.

Let us consider two DL concepts S and D (with S a supply description and D a demand description); if their conjunction $S \sqcap D$ is unsatisfiable in the TBox $\mathcal{T}$ representing the ontology, our aim is to retract requirements in D to obtain a concept K (for Keep) such that $K \sqcap S$ is satisfiable in $\mathcal{T}$.

Intuitively, the supplier is "weakening" his/her requests, to investigate whether what is left of the original request is still worth an interest. Clearly, a user is interested in *what* s/he must trade to initiate the transaction – a concept G (for Give up) such that D was made by G and K, that is, $D \equiv G \sqcap K$.

We stress that every argument in what follows could be restated exchanging supplier and demander, depending on who is actively starting the search.

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

We note that there is always the trivial solution $\langle G, K \rangle = \langle D, \top \rangle$ to a CCP. This solution corresponds to the most drastic contraction, that gives up everything of D. In our e-commerce setting, it would model the (infrequent) situation in which, in front of some very appealing counteroffer S, incompatible with mine, I just give up completely my specifications D in order to meet S.

On the other hand, when $S \sqcap D$ is satisfiable in $\mathcal{T}$, the "best" possible solution is $\langle \top, D \rangle$, 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 $contract(S, D, \mathcal{T})$ [19] to compute a minimal G (and a maximal K) for a demand D with respect to a given supply S and a TBox $\mathcal{T}$.

Once contraction has been applied, and consistency between the supply and the demand has been regained, there is still the problem with partial specifications, that is, it could be the case that the supply – though compatible – does not imply the demand. Then, it is necessary to assess what should be hypothesized in the supply in order to start the transaction with the demand. We call this non-standard inference *Concept Abduction*, in analogy to Charles Peirce's Abduction [48,41].

Definition 2. Let $\mathcal{L}$ be a DL, S, D, be two concepts in $\mathcal{L}$, and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$, where both S and D are satisfiable in $\mathcal{T}$. A *Concept Abduction Problem* (CAP), identified by $\langle \mathcal{L}, S, D, \mathcal{T} \rangle$, is finding a concept $H \in \mathcal{L}$ such that $\mathcal{T} \models S \sqcap H \sqsubseteq D$, and moreover $S \sqcap H$ is satisfiable in $\mathcal{T}$. We call H a *hypothesis* about S according to D and $\mathcal{T}$.

Also for Concept Abduction, there exist algorithms [23,19] that can compute H for $\mathcal{ALN}$ concepts S, D and a simple-TBox $\mathcal{T}$. We also note that numerical version $rankPotential(S, D, \mathcal{T})$ of the algorithm exists [25], computing the number of concept names in a Concept Abduction H, thus providing a score to the similarity between supply and demand.

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

4. DL modeling of e-marketplaces

We start pointing out that using standard database techniques to model a marketplace, we would be obliged to completely align the attributes of the supply and the demand in order to evaluate a match. The alignment would require a complete flattening of any structure of the descriptions, requiring a closed-world assumption. If supplies and demands are simple names or strings, the only possible match would be identity, resulting in an all-or-nothing approach to matchmaking. Although effective for fixed technical domains, such an approach misses the fact that supplies and demands usually have some sort of structure in them. Such a structure could be exploited in order to evaluate "interesting" inexact matches. Vector-based techniques taken by classical Information Retrieval can be used, too, thus reverting matchmaking to similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker [40] or in LARKS [55]. Obviously, lack of document structure in descriptions would make matching only probabilistic and strange situations may ensue. A further possibility is to use sets of words to describe the structure of supplies and demands. Although this is not our approach, it is interesting to discuss set-based matching since it highlights some properties of matchmaking. Beyond pure identity, one could compute some set-based relations between supplies and demands, such as inclusion, partial overlap, cardinality of set difference, etc. For instance, consider the demand

$D = \{computer, Intel, threeStorageDevices\}$ and the supply

$S = \{computer, Intel, LCDmonitor, threeStorage Devices, noSoftware\}$.

Although S and D are not identical, the fact that D is included in S tells the demander that every constraint posed by D is fulfilled by S, hence – from the point of view of the demander dd – S completely satisfies D. However, note that explicitly stated constraints in D do not completely satisfy S, although they do not exclude S either: it can happen that, asking the demander to refine D, an exact match ensues.

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. In the following, we assume a simple e-marketplace model, whose graphical sketch is pictured in Fig. 1, where descriptions of demands/supplies are stored in a repository together with domain ontologies relative to specific marketplaces. As new descriptions (demands/supplies) are submitted to the marketplace they are evaluated by the matchmaking service, which provides best available matches, or leverages initiation of search for negotiation spaces.

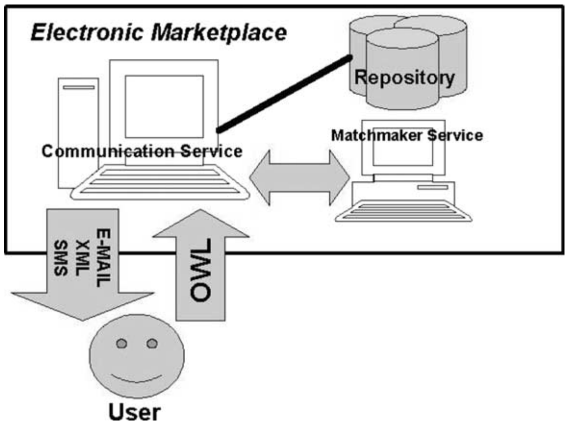


Fig. 1. E-Marketplace schematic architecture.

To simplify presentation of examples, we propose in Fig. 2 a toy ontology, which will be used as reference in what follows.

In an e-marketplace setting, a concept describing a demand, can be read as a set of constraints on the user request. For example, the simple demand $D = \mathtt{server} \sqcap (= 4\mathtt{hasCPU})$ can be represented by the set $\{\mathtt{server}, (= 4\mathtt{hasCPU})\}$ in which each element represents a constraint imposed by the user. We model a user description, representing a demand/supply in an e-marketplace, as two sets of constraints, accounting for negotiable and non-negotiable elements of the request description. In DL terms, we model the

$\mathtt{unix} \sqsubseteq \neg\mathtt{winX}$

$\mathtt{linux} \sqsubseteq \mathtt{unix}$

$\mathtt{CRTmonitor} \sqsubseteq \neg\mathtt{LCDmonitor}$

$\mathtt{USBpen} \sqsubseteq \mathtt{removableDevice}$

$\mathtt{AMD} \sqsubseteq \neg\mathtt{Intel}$

$\mathtt{computer} \sqsubseteq (\geq 1\ \mathtt{hasStorageDevice}) \sqcap (\geq 1\mathtt{hasComponent})$

$\mathtt{server} \equiv \mathtt{computer} \sqcap (\geq 2\ \mathtt{hasCPU})$

$\mathtt{personalComputer} \sqsubseteq \mathtt{computer}$

$\mathtt{personalComputer} \sqsubseteq \neg\mathtt{PDA}$

$\mathtt{homePC} \sqsubseteq \mathtt{personalComputer} \sqcap (= 1\ \mathtt{hasOS})$

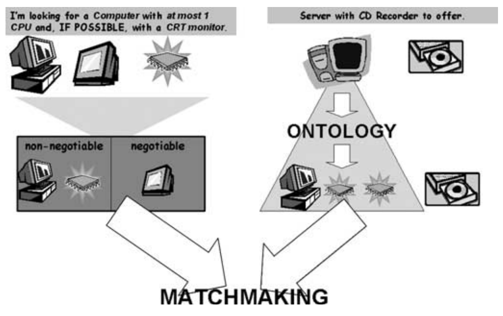
Fig. 2. The toy ontology used as reference in examples.

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 $\mathcal{ST}$, and we hence define $D = \mathcal{ST} \sqcap \mathcal{NG}$. A graphical sketch of the envisaged scenario is pictured in Fig. 3.

Obviously, if an element belongs to $\mathcal{NG}$ it cannot belong to $\mathcal{ST}$, i.e., if $(=4\texttt{hasCPU})$ 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 interaction between the ontology and the user's specifications about negotiable/non-negotiable constraints.

For example, consider the axiom $\texttt{server} \equiv \texttt{computer} \sqcap (\geqslant 2\texttt{hasCPU})$ in the reference ontology, and a user request $D = \texttt{server} \sqcap (\geqslant 2\texttt{hasCPU}) \sqcap (\geqslant 1\texttt{hasOS})$, with $\mathcal{ST} \equiv (\geqslant 2\texttt{hasCPU})$ and $\mathcal{NG} \equiv \texttt{server} \sqcap (\geqslant 1\texttt{hasOS})$. Based on the information contained in the previous axiom, and unfolding it in $\mathcal{NG}$, one obtains $(\geqslant 2\texttt{hasCPU})$ also in the negotiable constraints, clearly an incoherent specification of what is negotiable and what is not.

Considering $\mathcal{ST}$ and $\mathcal{NG}$ as conjunctions of $\mathcal{ALN}$ concepts, it is possible to check user negotiability specification coherence, using non-standard inference services in DL. This incoherency can be detected by checking if there exists an overlap between $\mathcal{ST}$ and $\mathcal{NG}$. Using Concept Abduction [23] or its numerical version [25], it is possible to compute how dissimilar $\mathcal{NG}$ is from $\mathcal{ST}$ as we will show in the following. Obviously, if $\mathcal{ST}$ and $\mathcal{NG}$ overlap, the user can be asked to reconsider negotiable/strict constraints.


I'm looking for a Computer with at most 1 CPU and, IF POSSIBLE, with a CRT monitor.
Server with CD Recorder to offer.
non-negotiable
negotiable
ONTOLOGY
MATCHMAKING
Fig. 3. The matchmaking scenario.

5. Finding negotiation spaces

As discussed in [25], if $D \sqcap S$ is unsatisfiable in $\mathcal{T}$, and the demander is willing to *retract* some of D's constraints, *partially matching* supplies could be reconsidered. Using both the notion of Concept Contraction and the logical formalization of negotiable constraints, we model negotiation spaces within a semantic-based matchmaking framework. Hence, in the following we assume that an actor in the marketplace is willing to play an active role, i.e., s/he is willing to consider retracting on some of the constraints expressed in the initial description to leverage negotiation.

With reference to the ontology in Fig. 2, let us consider a user demand $D = \texttt{homePC} \sqcap \forall\texttt{hasComponent.LCDmonitor}$, with $\mathcal{NG} = \forall\texttt{hasComponent.LCDmonitor}$ and $\mathcal{ST} = \texttt{homePC}$, and a supply $S = \texttt{homePC} \sqcap \forall\texttt{hasComponent.CRTmonitor}$. It is easy to verify that $D \sqcap S$ is unsatisfiable, hence a *partial match* ensues.

Solving a *CCP* we obtain a $\langle G,K \rangle$, where $G = \forall\texttt{hasComponent.LCDmonitor}$ and $K = \texttt{homePC}$.

By definition $K \sqcap S$ is satisfiable, hence, K potentially matches S. We also know that the demand D is a conjunction of $\mathcal{NG}$ and $\mathcal{ST}$.

Let us point out that, having $D \sqcap S$ unsatisfiable, if $\mathcal{ST} \sqcap S$ is satisfiable, then the unsatisfiability is due to $\mathcal{NG}$ (or to the conjunction of elements $\mathcal{ST}$ and other elements in $\mathcal{NG}$), i.e., the part of the demand the user is less interested in and may be willing to negotiate on, and – may be – retract.

Even if S has some constraints that are in conflict with D, we obtain that there is at least a part of $\mathcal{ST}$ (that is the most important one from the user's point of view), which can be potentially satisfied by S and, actually, *potentially* ranked if the user decides to give up a portion of his/her request.

If $\mathcal{ST}$ and S are unsatisfiable, there is no way to continue the search of a match, unless the active actor reformulates her/his request or the negotiable preferences. In the latter case, it is possible to suggest to the actor which part of $\mathcal{ST}$ has to be transformed into a negotiable constraint, by solving a *CCP* on $\mathcal{ST}$ and S. Any pair $\langle G, K\rangle$ can be interpreted as the part that must be set as negotiable (G), and the one remaining strict (K), in order to continue with the process. In fact, by definition, $G \sqcap S$ is unsatisfiable and $K \sqcap S$ is satisfiable.

Actually, the above scenario keeps its significance also if we flip over D and S, i.e., if we have a supplier expressing $\mathcal{ST}$ and $\mathcal{NG}$.

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

- A: description of the active actor (either a demand D or a supply S), with $A = \mathcal{ST} \sqcap \mathcal{NG}$.
- P: conversely defined description (with respect to the above item, either S or D).
- O: the ontology describing the marketplace domain.

```
**Algorithm** *spacesFinder*($P, A, O$)
**input** $\mathcal{ALN}$ concepts $P$, $A$, where $A = \mathcal{ST} \sqcap \mathcal{NG}$
**output** $\langle N_D, n, m\rangle$
**begin** = **algorithm**
1:  **if** ($A \sqcap P$ is unsatisfiable)
2:    **if** ($\mathcal{ST} \sqcap P$ is satisfiable){
3:      $\langle G, K\rangle = contract(P, A, O)$;
4:      $m = rankPotential(\top, G, O)$;
5:      $n = rankPotential(P, K, O)$;
6:      $N_K = rankPotential(\top, K, O)$;
7:      **return** $\langle N_K, n, m\rangle$;
8:    }
9:    **else**{
10:     Ask the active actor to change preferences on strict constraints;
11:     **if** YES {
12:       $\langle G_{\mathcal{ST}}, K_{\mathcal{ST}}\rangle = contract(P, \mathcal{ST}, O)$;
13:       $\mathcal{ST}_{new} = K_{\mathcal{ST}}$;
14:       $\mathcal{NG}_{new} = \mathcal{NG} \sqcap G_{\mathcal{ST}}$;
15:       $A_{new} = \mathcal{NG}_{new} \sqcap \mathcal{ST}_{new}$;
16:       **return** *spacesFinder*($P, A_{new}, O$);
17:     }
18:     **else return** $\langle -, \infty, \infty\rangle$;
19:   }
20: **else**{
21:   $n = rankPotential(P, A, O)$;
22:   $N_K = rankPotential(\top, A, O)$;
23:   **return** $\langle N_K, n, 0\rangle$;
24: }
**end algorithm**
```

We would like to point out that if the active actor is interested in knowing the constraints s/he has to negotiate it is sufficient having *spacesFinder* return also G computed as in row 3. A similar remark is sound also if the user is asked to reformulate the request. It is possible to ask the active actor if she/he is willing to negotiate also on $G_{\mathcal{ST}}$, computed in row 12.

Notice that in row 3, *spacesFinder* solves a *CCP* on P and A rather than on P and $\mathcal{NG}$.

The rationale is easily understandable using an example. Suppose to have the demander as the active actor, with $A = \texttt{computer} \sqcap (\geqslant 1\texttt{hasOS}) \sqcap \forall\texttt{hasOS.unix}$ and the seller as the passive one with $P = \texttt{computer} \sqcap \forall\texttt{hasOS.winX}$. Suppose also $\mathcal{NG} = \forall\texttt{hasOS.unix}$ and $\mathcal{ST} = (\geqslant 1\texttt{hasOS})$, which can be read as "I want a computer with a Unix Operating System already installed. I would accept also a computer with any O.S. pre-installed". Checking consistency, we obtain $A \sqcap P$ unsatisfiable, $\mathcal{ST} \sqcap P$ satisfiable and $\mathcal{NG} \sqcap P$ satisfiable, too. Then the inconsistency arises because of the combination of constraints both in $\mathcal{ST}$ and $\mathcal{NG}$. In order to make the request and the offer compatible, either $(\geqslant 1\texttt{hasOS})$ or $\forall\texttt{hasOS.unix}$ can be given up. Using a minimality criterion for *CCP*, which does not allow to retract on concepts in $\mathcal{ST}$ we are sure that *spacesFinder* in row 3 always computes a G not containing any concepts in $\mathcal{ST}$. Then, in our example we get $G = \forall\texttt{hasOS.unix}$ rather than $G = (\geqslant 1\texttt{hasOS})$.

Also notice that *spacesFinder* calls the *rankPotential* algorithm [25], which is used to compute the length of a concept, based on the corresponding ontology, as follows.

The call to *rankPotential*($\top, C, O$) computes the length of the $CAP : \top \sqcap H \sqsubseteq C$. A solution of such a *CAP* is $H = C$; then computing the length of H amounts to computing C length.

The algorithm *spacesFinder* allows to evaluate how promising is a match and to compare the results of a match between a request and several offers, returning the 3-tuple $\langle N_K, n, m\rangle$ of integers:

- N_K, the length of K that belongs to a solution of the *CCP* on P and A;
- n, the result of *rankPotential* on K and P, i.e., how dissimilar is P from K;[4]
- m, the length of G that belongs to a solution of the *CCP* on P and A.

The 3-tuple $\langle -, \infty, \infty\rangle$ 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 negotiate on. $n = \infty$ is a level of "unrecoverable" mismatch. We would like to point out that it may look sufficient, at a first glance, $m = |A|$, from now on N. Nevertheless this result corresponds to a distance between A and P equal to N, yet $\mathcal{ST} \sqcap P$ is still satisfiable. Having $n = 0$, $m = N$ is still possible when $\mathcal{ST} = \top$, and hence a *give up* is still possible on all elements of A.

Let us now describe the rationale of the three parameters with the help of some examples. We consider the following demand A and supply P:

$A = \texttt{personalComputer} \sqcap \forall\texttt{hasCPU.AMD}$

$P = \texttt{PDA} \sqcap \forall\texttt{hasCPU.Intel} \sqcap (\geqslant 1\texttt{hasCPU})$

$A \sqcap P$ is unsatisfiable because of the axiom on `personalComputer` in the ontology. Representing with $\mathcal{NG}_\infty$ and $\mathcal{ST}_\infty$ the corresponding sets of negotiable and non-negotiable concepts when the user discards P, i.e., *spacesFinder* returns $n = \infty$, and with $\mathcal{ST}_N$ and $\mathcal{NG}_N$ the set of negotiable and non-negotiable concepts when $n = N$, we obtain:

Case ∞. $\mathcal{ST}_\infty \sqcap P$ is unsatisfiable, i.e., P is in conflict with some constraints that the user is not willing to negotiate on.

$\mathcal{ST}_\infty = \texttt{personalComputer}$

$\mathcal{NG}_\infty = \forall\texttt{hasCPU.AMD}$

Then P has to be discarded.

Case N. The unsatisfiability depends on the negotiable elements of A.

$\mathcal{ST}_N = \top$

$\mathcal{NG}_N = \texttt{personalComputer} \sqcap \forall\texttt{hasCPU.AMD}$ The algorithm *spacesFinder*(P, A) returns the following result:

$G = \texttt{personalComputer} \sqcap \forall\texttt{hasCPU.AMD}$

$K = \top$

$m = 4 = |A| = N$; $n = 0$;

$N_K = 0$

P can be still considered among supplies potentially satisfying A, but it will probably have a bad score in a ranked list.

It is worth noticing that all the examples shown so far compute the length of a concept by counting specific constructs that appear in the concept definition. Anyway, we can assign, for each atomic concept and for each role, a weight that measures the relevance of the item both in the reference ontology and in the marketplace.

Having obtained elements to measure the quality of the match, the aim is to determine a single expression valid for all the cases. Given the following parameters:

$\frac{N_K}{N}$ is the fraction of N that can be negotiated to continue the transaction;

$\frac{m}{N_K}$ is the fraction of N_K that we have to negotiate to complete the transaction;

$\frac{n}{N_K}$ is the fraction of N_K that is not specified in P

We propose a function $U(N, N_K, n, m)$ that depends on the previous parameters, to measure how promising is a match also in the presence of both negotiable and strict constraints. In particular we defined and carried out experiments adopting the following simple closed form:

$$U(N, N_K, n, m) \approx \left|1 - \frac{N}{N-m}\left(1 - \frac{n}{N_K}\right)\right|.$$

Both terms $\frac{N}{N-m}$ and $\left|\left(1 - \frac{n}{N_K}\right)\right|$ have values $\in [1, \infty]$. In the former case, the lower bound is reached when $m = 0$, i.e., there is no negotiation of constraints in $\mathcal{NG}$ and the upper bound represents $N - m = 0$, i.e., the whole request has to be negotiated in order to continue with the process. In the latter case the lower bound identifies a subsumption relation between K and S.

Notice that the meaning of ∞ computed by $U(N, N_K, n, m)$ is completely different from the

[4] If the user is interested not only in "how dissimilar is" but also in "why" it is dissimilar, it is enough to solve the *CAP*: $P \sqcap H \sqsubseteq K$. In such a *CAP*, H represents a reason for dissimilarity.

one which can be computed by *spacesFinder*(P, A, O). While in the former case such a value represents a high position in a ranked list, in the latter one it represents that P has to be discarded and then not proposed to the user among the results.

5.1. Example behavior

We present here an illustrative example to better clarify the algorithm behavior. Let us consider the following descriptions:

S_1. Server with only Unix OS installed, equipped with a CRT monitor. Removable USB disk included.

$S_1 = \texttt{server} \sqcap \forall \texttt{hasOS.unix} \sqcap \forall \texttt{hasComponent.CRTmonitor} \sqcap \forall \texttt{hasStorageDevice.removableDevice} \sqcap (= 1\ \texttt{hasOS})$

S_2. PDA with Linux on-board and LCD monitor. Multimedia card removable storage device included.

$S_2 = \texttt{PDA} \sqcap \forall \texttt{hasOS.linux} \sqcap \forall \texttt{hasComponent.LCDmonitor} \sqcap \forall \texttt{hasStorageDevice.removableDevice}$

S_3. Single processor computer equipped with an high level LCD monitor.

$S_3 = \texttt{computer} \sqcap \forall \texttt{hasComponent.}(\texttt{LCDmonitor} \sqcap (\geqslant 1 \texttt{hasLevel}) \sqcap \forall \texttt{hasLevel.high}) \sqcap (\leqslant 1 \texttt{hasCPU})$

S_4. Personal computer for domestic use.

$S_4 = \texttt{homePC}$

and as a request:

D. I'm looking for a PC for domestic use, with Linux installed, I would prefer single processor computer, a CRT monitor and USB pen for data storing.

$D = \texttt{homePC} \sqcap \forall \texttt{hasOS.linux} \sqcap (\leqslant 1 \texttt{hasCPU}) \sqcap \forall \texttt{hasComponent.LCDmonitor} \sqcap \forall \texttt{hasStorageDevice.USBpen}$

with:

$\mathcal{ST} = \texttt{homePC} \sqcap \forall \texttt{hasOS.linux}$

$\mathcal{NG} = \forall \texttt{hasComponent.LCDmonitor} \sqcap (\leqslant 1 \texttt{hasCPU}) \sqcap \forall \texttt{hasStorageDevice.USBpen}$

Checking the consistency of marketplace items with the demand we obtain that S_1, S_2 and S_3 are not consistent with D, while S_4 and D are consistent w.r.t. each other.

With reference to the ontology in Fig. 2 (a concept length takes into account also the information modeled in the ontology), *spacesFinder* returns:$\langle 11, 4, 2\rangle$ for S_1, $U(N, N_K, n, m) = 0.25$

$\langle -, \infty, \infty\rangle$ for S_2 with no value for $U(N, N_K, n, m)$. We consider the circumstance where the active user does not want to retract on her/his non-negotiable constraints.

$\langle 12, 7, 1\rangle$ for S_3, $U(N, N_K, n, m) = 0.55$

$\langle 13, 6, 0\rangle$ for S_4, $U(N, N_K, n, m) = 0.46$

Then the ranked list returned as result has three items in the following order (from the best match to the worst): [S_1, S_4, S_3].

5.2. System and experiments

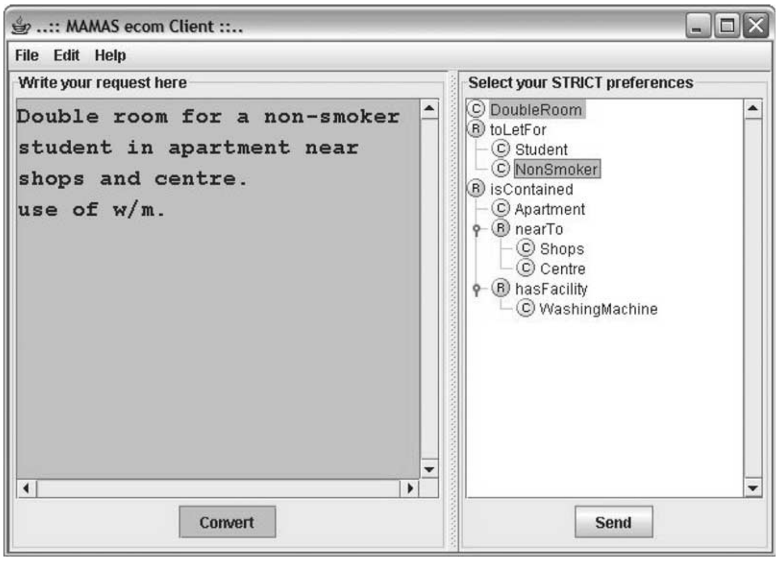
A simple prototype has been implemented for testing purposes. Users can express advertisements through a natural language interface. The system has been designed within the MAMAS framework and implements a natural language parser that translates demand/supply advertisements into structured DL expressions, automatically mapping sentences with concepts and roles of an ontology. The ontology has to be specific for each domain [20].

In order to evaluate the proposed approach, we have carried out a simple experiment, without any claim of completeness, due to the lack of ground truth measures. Hence we rely on a comparison of the system behavior versus the judgment of human volunteering users.

The domain we selected for our experiment is the students shared apartments rental, where we could rely on a fairly comprehensive ontology.

We examined several advertisements both from local newspapers and message boards, and selected 100 of them.

We asked 10 volunteering students – who were actually looking for a lodging – to place their demands, expressing both their strict constraints and negotiable ones using a graphical representation, provided by the system interface, of the sentence mapped in its corresponding DL model.



```
Double room for a non-smoker
student in apartment near
shops and centre.
use of w/m.
```

Fig. 4. The user interface used for experiments.

Fig. 4 shows a snapshot of the interface. Each user writes down his/her advertisement. When it has been parsed by the system, s/he can explicit his/her strict preferences; obviously unmarked preferences are assumed as negotiable.

The 20 higher ranking advertisements proposed by the system were presented to each volunteer randomly, and we asked him/her to sort them according to his/her judgment. The system-provided and user's rankings were then evaluated together using the R_{norm} [9] as quality measure of the accordance between the two lists.[5] The R_{norm} averaged on the whole set resulted 0.81, with a minimum of 0.67 and a maximum of 1.0. Largest part of disagreement was in the lower half of the rankings. The averaged R_{norm} resulting from the application of the approach proposed in [26] – which does not distinguish between strict and negotiable constraints – to the same dataset was 0.72. It should be noted that there is room for further enhancement, e.g., we may weight differently concepts to reflect the importance users give to specific items. Nevertheless we remark that this is still preliminary evaluation, and we are in the process of setting up a framework were users, based on the explanation provided by the system on the mismatch [22], can reformulate their queries dynamically.

[5] R_{norm} values are in the range [0, 1], with 1 corresponding to a system-provided ranking that is identical to the one provided by the human user, while lower values correspond to a proportional disagreement between the two.

6. Related work

Earliest matchmakers, based on the KQML, were proposed in [31,40]. In these works, matchmaking was introduced as an approach whereby potential producers/consumers could provide descriptions of their products/needs, either directly or through agents mediation, to be later unified by a matchmaker engine to identify potential matches. Nevertheless the proposed solutions to this challenging issue reverted to either a rule-based approach using the Knowledge Interchange Format (KIF) [33] (the SHADE [40] prototype) or a free text comparison (the COINS [40] prototype), which basically deals with descriptions as free-text retrieval tools do. Standard Information retrieval techniques have been also used in the recently proposed GRAPPA matchmaking framework [58].

Approaches similar to the cited ones were deployed in SIMS [1], which used KQML and LOOM as description language and InfoSleuth [37], which adopted KIF and the deductive database language LDL++. LOOM is also at the basis of the subsumption matching addressed in [34].

More recently, there has been a growing interest toward matchmaking engines and techniques, with emphasis placed either on e-marketplaces or generic Web services. Significant examples include [55,47] where a language, LARKS, is proposed specifically designed for agent advertisement. The matching process is carried out through five progressive stages, going from classical IR analysis of text to semantic match via Θ-subsumption. The notion, inspired by Software Engineering, of *plug-in* match is introduced to overcome in some way the limitations of a matching approach based on exact match. No ranking is presented but for what is called relaxed match, which basically reverts again to a IR free-text similarity measure. So a basic service of a semantic approach, such as inconsistency check, seems unavailable with this type of match.

In [27], an initial setting for semantic-based matchmaking was presented in a person-to-person framework, implemented using the CLASSIC system and based on subsumption matchmaking. In [35,57] a matchmaking framework was proposed, which operated on service descriptions in DAML + OIL and was based on the FaCT reasoner. An extension to the approach in [47] was proposed in [42] where two new levels for service profiles matching are introduced. JADE agent platform for Semantic web services discovery is used there, on a test ontology based on DAML-S. Notice that there, the *intersection satisfiable* level is introduced, whose definition is close to the one of *potential matching* proposed in [24]. The approach presented does not introduce a ranking method to measure proximity of service descriptions.

Semantic service discovery via matchmaking in the Bluetooth [8] framework was investigated in [52]. 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.

In [24,26] 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 [11] 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 [24] is presented in [18].

Also various current commercial electronic marketplaces try to provide some matchmaking capabilities between demand and supply. Jango [29] provides a system that basically only allows comparison, in terms of price, of goods available in on-line stores on the Internet. Obviously, the description of the product to be matched has to be complete and consistent and no reasoning on set containment or inconsistency check can be carried out. PersonaLogic [49] allows customers to impose constraints for alternatives seeking. It must be pointed out that constraints cannot be dynamically placed but have to be taken from a pre-determined category set. Kasbah [39] is a more effective system, which allows to dynamically set constraints, yet it does not allow handling of inconsistency and partial or potential matches. A similar approach is also deployed in Tete-a-Tete [44].

A more advanced constraint based approach is proposed in [38], able to handle conflicting preferences in demands/supplies. Consistency check of preferences is accomplished visiting an offer synthesis graph with path consistency algorithm each time a new offer is entered. A further example is Smartclient [50], a system that allows users basic criteria adjustment, by presenting an interface that shows the initial search space, which can be reduced by further user interaction with the results. The underlying system basically relies on partial constraint satisfaction techniques. A recent proposal along the same lines is in [59] where negotiation agents are formally modeled using an object-oriented constraint language.

IBM's Websphere (SilkRoad) matchmaking environment was the first example of commercial solution that places an explicit emphasis on the matchmaking between a demand and a supply in a peer-to-peer way, which is referred to in [36] as *symmetric* matchmaking. The environment is based on a matchmaking engine that describes supplies/demands as properties and rules. Properties are name-value pairs constructed using an extension of Corba Trading service language. Rules are basically constructed using a generic script language. Matching is then accomplished by simply comparing properties and verifying rules. No notion of distinction between total, partial, potential and inconsistent matches are present.

A similar approach, with descriptions defined in XML and again a rule based decision system is in [16]. Here descriptions of supply/demand can be stored in the e-service platform when a match is not available for further processing should a counterpart become available. In [53], an extension to the original Websphere matchmaker is proposed, which introduces users' specification of negotiable constraints when no total match is available. So the approach aims at some of the issues also addressed in this paper, but with a different, constraint based, perspective.

6.1. Related non-standard inferences

In [7,6], the Difference Operator in DLs, originally devised in [56], was proposed for matchmaking in the framework of web services. The approach uses the Concept Difference, followed by a set covering operation optimized using hypergraph techniques. The adopted DL is 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. Furthermore, 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 [13], an algorithm is proposed for Difference on approximation of concepts. Notice that Concept Abduction may appear similar to Concept Difference, yet it is not so. In [56] difference is expressed as: $D - C = \max_{\sqsubseteq}\{E \in \mathscr{L} : (C \sqcap E) \equiv D\}$.

In [13], it is expressed as: $D - C = \min_{\prec}\{E \in \mathscr{L} : (C \sqcap E) \equiv D\}$. Yet it should be noticed that $D \sqsubseteq C$ is needed in the difference; without it $D - C$ would result in a subminimal solution of $\langle \mathscr{L}, C, D, T \rangle$.

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 by *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 bring to loss of information. For example having $S = \neg A \sqcap B$ and $D = A \sqcap C$, we obtain $lcs = \top$. There is no way to recover information of B in S and of C in D. Matching in DLs has been widely treated in [5] although with no relation to matchmaking. In fact, in that work expressions denoting concepts are considered, with variables in expressions. Then a match is a substitution of variables with expressions that makes a concept expression equivalent to another. Also the more general setting of concept rewriting in DLs has no direct relation with matchmaking.

7. Conclusion

The recent interest toward DLs, which are endowed of several useful characteristics and are currently one of the cornerstones of the Semantic web initiative, has led to a flourishing of proposals enriching the matchmaking process between demand supply with semantics, at first only based on classical DLs inference services. Motivated by the observation that the matchmaking process has to be modeled well taking into account commonsense reasoning, we noted that, when no perfect match exists, and individuals are interested in bargaining, they tend to look for promising – though not perfect – matches. They may then be willing to partially retract their original requirements, or simply told "negotiate". This dynamic, and reasonable, process cannot receive much benefit by standard DLs inference services, which usually provide boolean type answers. In this paper we presented novel logic-based approach and algorithms to find negotiation spaces between demand and supply, and to rank matches, adopting a closed-form function, according to how much promising, though inexact, they are.

To this purpose we exploited recently proposed non-standard inference services in DLs, Concept Contraction and Concept Abduction. A simple set of experiments, carried out with the aid of volunteers, confirmed the validity of the approach with respect to commonsense reasoning.

Future work is aimed at the extension of the logic adopted – while trying to keep the problems tractable, and to modeling an approach for implicit knowledge elicitation from a set of descriptions in a marketplace, to be used in users requests refinement.

Acknowledgements

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

References

[1] Y. Arens, C.A. Knoblock, W. Shen, Query reformulation for dynamic information integration, Journal of Intelligent Information Systems 6 (1996) 99–130.
[2] F. Baader, Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles, in: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI2003), 2003, pp. 319–324.
[3] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, P. Patel-Schneider (Eds.), The Description Logic Handbook, Cambridge University Press, 2002.
[4] F. Baader, I. Horrocks, U. Sattler, Description logics as ontology languages for the semantic web, in: D. Hutter, W. Stephan (Eds.), Festschrift in honor of Jörg Siekmann, Lecture Notes in Artificial Intelligence, Springer-Verlag, 2003.
[5] F. Baader, R. Kusters, A. Borgida, D. Mc Guinness, Matching in description logics, Journal of Logic and Computation 9 (3) (1999) 411–447.
[6] B. Benatallah, M.-S. Hacid, C. Rey, F. Toumani, Request rewriting-based web service discovery, in: International Semantic Web ConferenceLecture Notes in Computer Science, vol. 2870, Springer, 2003, pp. 242–257.
[7] B. Benatallah, M.-S. Hacid, C. Rey, F. Toumani, Semantic reasoning for web services discovery, in: Proceedings of Workshop on E-Services and the Semantic Web at WWW 2003, May 2003.
[8] Bluetooth. Available from: <http://www.bluetooth.com>.
[9] P. Bollmann, F. Jochum, U. Reiner, V. Weissmann, H. Zuse, The LIVE-Project-Retrieval experiments based on evaluation viewpoints, in: Proceedings of the 8th Annual International ACM/SIGIR Conference on Research and Development in Information Retrieval, ACM, New York, 1985, pp. 213–214.
[10] A. Borgida, Description logics in data management, IEEE Transactions on Knowledge and Data Engineering 7 (5) (1995) 671–682.
[11] A. Borgida, P.F. Patel-Schneider, A semantics and complete algorithm for subsumption in the CLASSIC description logic, Journal of Artificial Intelligence Research 1 (1994) 277–308.
[12] R. Brachman, H. Levesque, The tractability of subsumption in frame-based description languages, in: Proceedings of the Fourth National Conference on Artificial Intelligence (AAAI-84), Morgan Kaufmann, Los Altos, 1984, pp. 34–37.
[13] S. Brandt, R. Küsters, A.-Y. Turhan, Approximation and difference in description logics, in: Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning (KR2002), Morgan Kaufman, 2002, pp. 203–214.
[14] A. Calì, D. Calvanese, G.D. Giacomo, M. Lenzerini, Data integration under integrity constraints, Information Systems 29 (2) (2004) 147–163.
[15] D. Calvanese, G. De Giacomo, M. Lenzerini, On the decidability of query containment under constraints, in: Proceedings of the Seventeenth ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98), 1998, pp. 149–158.
[16] F. Casati, M.C. Shan, Dynamic and adaptive composition of e-services, Information Systems 26 (2001) 143–163.
[17] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, 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.
[18] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello. Logic based approach to web services discovery and matchmaking, in: Proceedings of the E-Services Workshop at ICEC'03, September 2003.
[19] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, 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.

[20] S. Coppi, T. Di Noia, E. Di Sciascio, F.M. Donini, A. Pinto, Ontology-based natural language parser for e-marketplaces18th International Conference on Industrial and Engineering Applications of Artificial Intelligence, 3533, Springer-Verlag, 2005, pp. 279–289.

[21] DAML + OIL Specifications. Available from: <www.daml.org/2001/03/daml+oil-index.html>, 2001.

[22] T. Di Noia, E. Di Sciascio, F. Donini, Extending Semantic-based matchmaking via concept abduction and contraction, in: Proceedings of the 14th International Conference on Knowledge Engineering and Knowledge Management (EKAW 2004)Lecture Notes in Artificial Intelligence, 3257, Springer, 2004, pp. 307–320.

[23] T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello, Abductive matchmaking using description logics, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003), Morgan Kaufmann, Los Altos, 2003, pp. 337–342. Acapulco, Messico, August 9–15.

[24] T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello, Semantic matchmaking in a P-2-P electronic marketplace, in: Proc. Symposium on Applied Computing (SAC '03), ACM, 2003, pp. 582–586.

[25] T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello, A system for principled Matchmaking in an electronic marketplace, in: Proceedings of the International World Wide Web Conference (WWW '03), ACM, New York, 2003, pp. 321–330, Budapest, Hungary, May 20–24.

[26] T. Di Noia, E. Di Sciascio, F. Donini, M. Mongiello, A system for principled matchmaking in an electronic marketplace, International Journal of Electronic Commerce 8 (4) (2004) 9–37.

[27] E. Di Sciascio, F. Donini, M. Mongiello, 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.

[28] F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf, Reasoning in Description Logics, in: G. Brewka (Ed.), Principles of Knowledge Representation, Studies in Logic, Language and Information, CSLI Publications, 1996, pp. 193–238.

[29] R. Doorenbos, O. Etzioni, D. Weld, A scalable comparison-shopping agent for the world-wide web, in: Proceedings of the International Conference on Autonomous Agents '97, ACM, 1997, pp. 39–48.

[30] M. Dumas, B. Benatallah, N. Russell, M. Spork, A configurable matchmaking framework for electronic marketplaces, Electronic Commerce Research and Appplications 3 (1) (2004) 95–106.

[31] T. Finin, R. Fritzson, D. McKay, R. McEntire, KQML as an agent communication language, in: Proceedings of the Third International Conference on Information and Knowledge Management (CIKM'94), ACM, 1994, pp. 456–463.

[32] P. Gärdenfors, Knowledge in Flux: Modeling the Dynamics of Epistemic States, Bradford Books, MIT Press, Cambridge, MA, 1988.

[33] M.R. Genesereth, Knowledge interchange format, in: Principles of Knowledge Representation and Reasoning: Proceedings of the 2nd International Conference, Cambridge, MA, Morgan Kaufmann, Los Altos, 1991, pp. 599–600.

[34] Y. Gil, S. Ramachandran, PHOSPHORUS: a task based agent matchmaker, in: Proc. International Conference on Autonomous Agents '01, ACM, 2001, pp. 110–111.

[35] J. Gonzales-Castillo, D. Trastour, C. Bartolini. Description logics for matchmaking of services, in: Proceedings of the KI-2001 Workshop on Applications of Description Logics (ADL-2001), volume 44 of CEUR Workshop Proceedings, 2001.

[36] Y. Hoffner, C. Facciorusso, S. Field, A. Schade, Distribution issues in the design and implementation of a virtual market place, Computer Networks 32 (2000) 717–730.

[37] N. Jacobs, R. Shea, Carnot and infosleuth – database technology and the web, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, ACM, 1995, pp. 443–444.

[38] N. Karacapilidis, P. Moraitis, Building an agent-mediated electronic commerce system with decision analysis features, Decision Support Systems 32 (2001) 53–69.

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

[40] D. Kuokka, L. Harada, Integrating information via matchmaking, Journal of Intelligent Information Systems 6 (1996) 261–279.

[41] H. Levesque, A knowledge-level account for abduction, in: Proceedings of the Eleventh International Joint Conference on Artificial Intelligence (IJCAI'89), Morgan Kaufmann, Los Altos, 1989, pp. 1061–1067.

[42] L. Li, I. Horrocks, A software framework for matchmaking based on semantic web technology, International Journal of Electronic Commerce 8 (4) (2004).

[43] J. Madhavan, P. Bernstein, E. Rahm, Generic schema matching with cupid, in: Proceedings of the VLDB '01, 2001, pp. 49–58.

[44] P. Maes, R. Guttman, A. Moukas, Agents that buy and sell, Communications of the ACM 42 (3) (1999) 81–91.

[45] D. McGuinness, R. Fikes, J. Hendler, L. Stein, DAML + OIL: An ontology language for the semantic web, IEEE Intelligent Systems 17 (5) (2002) 72–80.

[46] OWL. Available from: <www.w3.org/TR/owl-features/>.

[47] M. Paolucci, T. Kawamura, T. Payne, K. Sycara, Semantic Matching of Web Services Capabilities, in: The Semantic Web – ISWC 2002Lecture Notes in Computer Science, 2342, Springer-Verlag, 2002, pp. 333–347.

[48] C. Peirce, Abduction and induction, in: J. Buchler (Ed.), Philosophical Writings of Peirce, 1955 (Chapter 11).

[49] PersonaLogic. Available from: <http://www.PersonaLogic.com>.

[50] P. Pu, B. Faltings. Enriching buyers' experience, CHI Letters 2 (1) (2000).

[51] Resource Description Framework. Available from: <http://www.w3.org/RDF/>.

[52] S. Avancha, A. Joshi, T. Finin, Enhanced service discovery in bluetooth, IEEE Computer (2002) 96–99.

[53] M. Ströbel, M. Stolze, A matchmaking component for the discovery of agreement and negotiation spaces in electronic markets, Group Decision and Negotiation 11 (2002) 165–181.

[54] K. Sycara, M. Paolucci, M. Van Velsen, J. Giampapa, The RETSINA MAS infrastructure, Autonomous agents and multi-agent systems 7 (2003) 29–48.

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

[56] 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), MK, 1994, 540–550.

[57] D. Trastour, C. Bartolini, C. Priest, Semantic web support for the business-to-business e-commerce life-cycle, in: Proceedings of the International World Wide Web Conference (WWW) '02, ACM, 2002, pp. 89–98.

[58] D. Veit, J. Muller, M. Schneider, B. Fiehn, Matchmaking for autonomous agents in electronic marketplaces, in: Proceedings of the International Conference on Autonomous Agents '01, ACM, 2001, pp. 65–66.

[59] H. Wang, S. Liao, L. Liao, Modeling constraint-based negotiating agents, Decision Support Systems 33 (2002) 201–217.