Mobile Semantic-Based Matchmaking: A Fuzzy DL Approach

Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio

Politecnico di Bari
via Re David 200, I-70125
Bari, Italy
{m.ruta,f.scioscia,disciascio}@poliba.it

Abstract. Novel wireless handheld devices allow the adoption of revised and adapted discovery approaches originally devised for the Semantic Web in mobile ad-hoc networks. Nevertheless, capabilities of such devices require an accurate re-design of frameworks and algorithms to efficiently support mobile users. The paper focuses on an implementation of concept abduction and contraction algorithms in (fuzzy) $\mathcal{ALN}(\mathrm{D})$ DL settings to perform semantic matchmaking and provide logical explanation services. OWL-DL Knowledge Bases have been properly exploited to enable standard and non-standard inference services. The proposed framework has been implemented and tested in a *fire hazards* prevention case study: early experimental results are reported.

1 Introduction

Current mobile devices are increasingly effective and powerful but anyway they maintain some limitations which have to be taken into account when designing systems and applications. The paper proposes a novel matchmaking framework suitable for resource-constrained devices and contexts where canonical discovery is unfeasible or even impractical. Particularly, we propose a revision of abduction and contraction algorithms presented in [1] in order to allow an exploitation of them in mobility. Note that, even if they have rising performances, battery powered handheld devices are networked by means of wireless low-throughput links and are basically unable to run heavy reasoning algorithms explicitly devised for PCs and servers over the Internet. Knowledge Representation techniques and approaches –which originally enhanced code-based discovery [2]– have been revised to be effectively suitable in volatile ubiquitous computing contexts by-passing the presence of fixed workstations.

The proposed framework and approach has been applied and tested in a mobile *fire risk prevention* case study even if it is cross-applicable to several scenarios. New approaches to disaster prevention focus on providing assistance to discover possible risks in given areas and activities before disasters happen. Thanks to a continuous surveillance of context parameters and taking into account a knowledge-based domain and risk modeling, possible hazards can be detected in order to quickly perform recovery procedures to take danger under

L. Aroyo et al. (Eds.): ESWC 2010, Part I, LNCS 6088, pp. 16–30, 2010.

control. In what follows an illustrative example is presented to better explain potentialities our approach provides and to let emerge its novelty. A performance examination referred to the case study has been also carried out allowing to evaluate system behavior.

We selected Description Logics (DLs) as reference logic formalism family [3] and particularly we referred to a sublanguage deriving from OWL DL[1], $\mathcal{ALN}$(D), to model ontologies and environmental annotations whereas the proposed system adopts DIG 2.0[2] as reference syntax. Noteworthy is the exploitation of fuzzy DL operators to better model vague real-world requirements in matchmaking issues.

The remaining of the paper is structured as follows: in the next Section we survey most relevant related work, Section 3 is devoted to present a background of the proposed approach; afterward, in Section 4 we move on to description of theoretical framework. Relevant features of proposed algorithms and architecture are better clarified in Section 5 with the aid of a simple illustrative case study whereas Section 6 reports on experimental evidences related to the implemented prototype. Finally conclusions close the paper.

2 Related Work

Most mobile reasoning engines currently provide only simple rule processing through forward/backward chaining. Interesting instances include mobile Prolog engines [4] –which are reduced version of PC engines– and the mobile composable reasoner in [5]. Their expressiveness, however, is too limited to support advanced applications such as semantic-based matchmaking and resource discovery.

Adapting tableaux algorithms –used in most "wired" DL reasoners– to mobile computing platforms may allow more expressive languages to be used, but efficient implementation of useful non-standard reasoning services is still an open problem due to resource limitations. Most optimization techniques [6] cannot be adopted in mobile systems, since they decrease running time but increase main memory usage.

Pocket KRHyper [7] was the first reasoning engine for mobile devices. It supports the Description Logic $\mathcal{ALCHIR}+$ and was built as a Java ME (Micro Edition) library. Pocket KRHyper was exploited by the authors in a DL-based matchmaking framework between user profiles and descriptions of mobile resources/services [8]. However, its limitations in size and complexity of managed logical expressions are very tight because "out of memory" errors are frequent. To overcome those limitations, Steller and Krishnaswamy [9] introduced a set of tableaux optimizations to reduce memory consumption. Proposed techniques were implemented in *mTableau*, a modified version of Pellet, a popular open-source reasoning engine for Java Standard Edition (SE). Comparative performance tests were performed on a PC, showing faster turnaround times than

[1] OWL Web Ontology Language, W3C Recommendation 10 February 2004, `http://www.w3.org/TR/owl-features/`

[2] DIG 2.0: The DIG Description Logic Interface, `http://dig.cs.manchester.ac.uk/overview.html`

both unmodified Pellet and Racer, another widely used reasoner in the Semantic Web. Nevertheless, Java SE as a technology is not tailored to the current generation of mobile computing devices, even though some experimental projects are trying to port Java SE to mobile and embedded devices. Both the above systems only support standard satisfiability and subsumption inference services. Consequently, they can only distinguish among full, potential and partial match types (adopting the terminology of [10]). This is due to the fact that satisfiability and subsumption provide only binary "yes/no" answers. Abductive matchmaking is needed to determine what part of a request is not fully satisfied by a supplied resource, thus enabling a more fine-grained semantic ranking as well as explanation of outcomes.

In [11], a different approach to adapt logic-based reasoning services to pervasive computing contexts was proposed. It was based on simplifying the underlying logical languages and admitted axioms in a KB, so that standard and non-standard reasoning services could be reduced to set-based operations. KB management and reasoning services were then executed through a data storage layer, based on a mobile RDBMS (Relational DBMS). A specialized database schema template allowed KB classification and inference algorithms to be performed through a small set of SQL queries. The proposed discovery model was suitable for pervasive contexts characterized by a large amount of "lightweight" resources with a low level of semantic sophistication. This approach was further investigated in [12], by increasing the expressiveness of the language and allowing larger ontologies and more complex descriptions, through the adoption of a mobile OODBMS (Object-Oriented DBMS). Though the approach has relevant benefits, a native language implementation can certainly grant greater flexibility in extending algorithms of the core reasoning engine to more expressive logical languages and in implementing additional inference services.

3 Background

In what follows most relevant basics for the proposed approach will be provided with specific reference to (fuzzy) Description Logics and matchmaking.

3.1 Description Logics

Description Logics (DLs) are a family of logic formalisms for Knowledge Representation [3]. Basic syntax elements are: *concept* names, *role* names, and *individuals*. They can be combined using *constructors* to build complex concept and role *expressions*. Each DL is featured by a different set of constructors. A constructor used in every DL is the one allowing the *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* and *universal role quantification*. Other constructs may involve counting, as *number restrictions*. Many other constructs can be defined, so increasing the expressiveness of the related DL but

also the computational complexity of related inference services [13]. *OWL-DL* is a language based on DLs theoretical studies which allows a great expressiveness keeping computational completeness and decidability.

In a DL framework, an ontology $\mathcal{T}$ (a.k.a. Terminological Box or TBox) is composed by a set of axioms in the form: $A \sqsubseteq D$ or $A \equiv D$ where A is an atomic concept and D is generic $\mathcal{ALN}$ concept. In particular, we call simple-TBox all those set of axioms such that if A appears in the left hand side (lhs) of a concept equivalence axioms then it cannot appear also in the lhs of any concept inclusion axiom. In this paper we refer to the $\mathcal{ALN}$(D) (Attributive Language with Unqualified Number Restrictions with Concrete Domains) subset of OWL-DL, which has polynomial computational complexity for standard and non-standard inferences in "bushy but not deep" TBoxes [14].

With respect to $\mathcal{ALN}$, $\mathcal{ALN}$(D) allows to model a set of *features* f having D as domain, and a set of unary predicates p in D. Each feature can be seen as: $p(f)$ where $f : \Delta \rightarrow R$. So the following relation ensues:

$$p(f)^{\mathcal{I}} = \{c \in \Delta \mid f(c) \in p^{D}\} \tag{1}$$

For instance, if we consider the integers concrete domain D, p can be represented by one of $\geq_k (\cdot)$, $\leq_k (\cdot)$ or $=_k (\cdot)$ operators, where k is a given value in the domain associated to f.

Hereafter, for compactness we will formalize examples by adopting DL syntax (instead of OWL-DL or its syntactic variant DIG [15]), whereas in our prototypes DIG is exploited because it is less verbose w.r.t. OWL.

3.2 Fuzzy DL

OWL DL becomes less suitable when facts to be represented do not have a precise definition. For instance, let us suppose we have to model the description of a "large fuel depot containing a lot of liquefied petroleum gas with various expert workers operating in the warehouse, and also accessible to non-authorized staff members" in a *fire hazard* domain. A plain translation into OWL DL is difficult, as it involves so called fuzzy or vague concepts, like "large", "a lot", "various", whose unambiguous definition is practically unfeasible. Hence we have to consider a fuzzy extension of $\mathcal{ALN}$(D) DL, and exploit its syntax and semantics.

Basically, a fuzzy set A w.r.t. a universe X is characterized by a membership function $\mu_A : X \rightarrow [0, 1]$ assigning an *A-membership degree*, $\mu_A(x)$, to each element x in X. $\mu_A(x)$ gives us an estimation of "how much" x belongs to A. Typically, if $\mu_A(x) = 1$, then x definitely belongs to A, while $\mu_A(x) = 0.5$ means that x is could be an element of A [16]. The notion of degree of membership $\mu_A(x)$ of an element $x \in X$ w.r.t. the fuzzy set A over X can also be seen as the degree of truth in $[0, 1]$ of the statement "x is A". Hence a *fuzzy set* A can be defined as set of ordered pairs composed by each X element along with the corresponding value for the membership function:

$$A = \{(x, \mu_A(x)) \mid x \in X\} \tag{2}$$

According to the specific applications, different membership functions can be exploited. The most frequent and widespread are *Trapezoidal*, *Triangular*, *Left shoulder* and *Right shoulder* functions (see [16] for details). They will be also used in the approach proposed here.

3.3 Matchmaking

Given an ontology $\mathcal{T}$ and two generic concepts C and D, DL matchmakers expose at least two basic standard reasoning services: concept **subsumption** and concept **satisfiability**. In a nutshell they can be defined as in the following:

Concept subsumption. Check if C is more specific than (implies) D with respect to the information modeled in $\mathcal{T}$. In formulae we write $\mathcal{T} \models C \sqsubseteq D$.

Concept satisifiability. Check if the information in C is not consistent with respect to the information modeled in $\mathcal{T}$. In formulae we write $\mathcal{T} \models C \sqsubseteq \bot$.

Subsumption and satisfiability may be powerful tools in case a Boolean answer is needed. However, in more advanced scenarios where the so called *Open World Assumption* (OWA) is made, *yes/no* answers do not provide satisfactory results. Often an outcome explanation is required. In [14] Concept Abduction Problem (CAP) $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$ was introduced and defined as non standard inference problem for DLs, to provide an explanation when subsumption does not hold. In a few words, given an ontology $\mathcal{T}$ and two concepts C and D in a DL $\mathcal{L}$, if $\mathcal{T} \models C \sqsubseteq D$ is `false` then we compute a concept H (for hypothesis) such that $\mathcal{T} \models C \sqcap H \sqsubseteq D$. That is, H is a possible explanation why resource features do not imply requested ones or, in other words, H represents missing capabilities in the resource C in order to completely satisfy a request D w.r.t. $\mathcal{T}$. Actually, given a CAP there is more than one valid solution. Some minimality criteria have to be defined. We refer the interested reader to [14] for further details.

If the conjunction $C \sqcap D$ is unsatisfiable in the TBox $\mathcal{T}$ representing the ontology, *i.e.*, C, D are not compatible with each other, the requester can retract some requirements G (for Give up) in D, to obtain a concept K (for Keep) such that $K \sqcap C$ is satisfiable in $\mathcal{T}$ (Concept Contraction Problem, CCP). CCP is formally defined as follows. Let $\mathcal{L}$ be a DL, C, D be two concepts in $\mathcal{L}$ and $\mathcal{T}$ be a set of axioms in $\mathcal{L}$, where both C and D are satisfiable in $\mathcal{T}$. A Concept Contraction Problem, identified by $\langle \mathcal{L}, C, D, \mathcal{T} \rangle$ 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 $K \sqcap C$ is satisfiable in $\mathcal{T}$. Then K is a contraction of D according to C and $\mathcal{T}$. Like for the abduction problem, some minimality criteria in the contraction must be defined [14], since usually one wants to give up as few things as possible.

4 Framework and Approach

Here implementation of algorithms featuring the matchmaking described above is analyzed. Each request and supply (from now on D and S) as well as the given TBox $\mathcal{T}$ will be expressed in $\mathcal{ALN}$(D) Description Logic[3].

[3] We refer the reader to [14,10] for exhaustive examples and wider argumentation.

4.1 Algorithms

Algorithm 4.1. Concept Contraction

Require: $\langle \mathcal{L}, D, S, \mathcal{T} \rangle$ with $\mathcal{L} = \mathcal{ALN}(D)$, acyclic $\mathcal{T}$
Ensure: $\langle G, K \rangle$ with concept G, K in $\mathcal{L}$
  if $D == \bot$ **then**
    return $\langle \bot, \top \rangle$
  else
    G := $\top$
    K := $\top \sqcap D$
    for all concept name A in K **do**
      U := $unfolding(\langle \mathcal{L}, A, \mathcal{T} \rangle)$
      for all concept name A in U **do**
        if there exists B in S such that $B = \neg A$ **then**
          G := $G \sqcap A$
          remove A from K
        end if
      end for
    end for
    for all concept $C := (\geq xR)$ in K **do**
      if there exists $(\leq yR)$ in S and $y < x$ **then**
        replace C with $(\geq yR)$
        $G := G \sqcap C$
      end if
      for all concept $\forall R.E$ in K **do**
        if there exists $\forall R.F$ in S **then**
          $\langle G', K' \rangle := Contract(\langle \mathcal{L}, E, F, \mathcal{T} \rangle)$
          $G := G \sqcap \forall R.G'$
          replace $\forall R.E$ in K with $\forall R.K'$
        end if
      end for
    end for
    for all concept $C := (\leq xR)$ in K **do**
      if there exists $(\geq yR)$ in S and $y > x$ **then**
        replace C with $(\leq yR)$
        $G := G \sqcap C$
      end if
    end for
    for all predicate $\geq_x g$ *in* K **do**
      if there exists $\leq_y g$ in S and $y < x$ **then**
        replace $\geq_x g$ with $\geq_y g$
        $G := G \sqcap C$
      end if
    end for
    for all predicate $\leq_x g$ in $K<$ **do**
      if there exists $\geq_y g$ in S and $y > x$ **then**
        replace $\leq_x g$ with $\leq_y g$
        $G := G \sqcap C$
      end if
    end for
  end if

  return $\langle G, K \rangle$

Algorithm 4.2. Concept Abduction

Require: $\langle \mathcal{L}, D, S, \mathcal{T} \rangle$ with $\mathcal{L} = \mathcal{ALN}(D)$, acyclic $\mathcal{T}$
Ensure: H in $\mathcal{L}$
  $H := \top$
  for all concept name A in D **do**
    if does not exist B in S such that $B \sqsubseteq A$ **then**
      $H := H \sqcap A$
    end if
  end for
  for all concept $(\geq xR)$ in D **do**
    if does not exist $(\geq_y R)$ in S with $y \geq x$ **then**
      $H := H \sqcap (\geq xR)$
    end if
  end for
  for all concept $(\leq xR)$ in D **do**
    if does not exist $(\leq yR)$ in S with $x \geq y$ **then**
      $H := H \sqcap (\leq xR)$
    end if
  end for
  for all concept $\geq_x g$ in D **do**
    if does not exist $\geq_y g$ in S with $y \geq x$ **then**
      $H := H \sqcap \geq_x g$
    end if
  end for
  for all concept $\leq_x g$ in D **do**
    if does not exist $\leq_y g$ in S with $x \leq y$ **then**
      $H := H \sqcap \leq_x g$
    end if
  end for
  for all concept $\forall R.E$ in D **do**
    if there exists $\forall R.F$ in S **then**
      $H' := Abduce(\langle \mathcal{L}, E, F, \mathcal{T} \rangle)$
      $H := H \sqcap \forall R.H'$
    else
      $H := H \sqcap \forall R.E$
    end if
  end for
  return H

The matchmaking starts when are available the unfolded versions of D and S, expressed in Conjunctive Normal Form (CNF) [14]. The *Unfolding* and *Normalization* procedures aiming to pre-process D and S are outlined later on. Given the $\mathcal{ALN}$(D) DL, an acyclic TBox $\mathcal{T}$ and a concept C in $\mathcal{ALN}$(D), the *unfolding* procedure reported hereafter produces a new concept equivalent to C w.r.t. $\mathcal{T}$.

$A \rightarrow A \sqcap C$ if $A \sqsubseteq C \in \mathcal{T}$
$A \rightarrow C$ if $A \equiv C \in \mathcal{T}$
$A \rightarrow A \sqcap \neg B_1 \sqcap \ldots \sqcap \neg B_k$ if $disj(A, B_1, \ldots, B_k) \in \mathcal{T}$

Furthermore, the *Conjunctive Normal Form* of a concept can be obtained by applying the following substitutions (after the unfolding procedure) until no more substitutions are possible:

$$
\begin{aligned}
C \sqcap \bot &\rightarrow \bot \\
(\geq n\, R) \sqcap (\leq m\, R) &\rightarrow \bot \text{ if } n > m \\
A \sqcap \neg A &\rightarrow \bot \\
(\geq n\, R) \sqcap (\geq m\, R) &\rightarrow (\geq n\, R) \text{ if } n > m \\
(\leq n\, R) \sqcap (\leq m\, R) &\rightarrow (\leq n\, R) \text{ if } n < m \\
\forall R.D_1 \sqcap \forall R.D_2 &\rightarrow \forall R.\,(D_1 \sqcap D_2) \\
\forall R.\bot &\rightarrow \forall R.\bot \sqcap (\leq 0\, R)
\end{aligned}
$$

For an unfoldable TBox $\mathcal{T}$ in $\mathcal{ALN}$(D) DL, any concept expression after normalization and unfolding can be expressed as the conjunction of five concept sets [11]: (i) all atomic concepts and negations; (ii) all universal quantifiers; (iii) all $\geq$ number restrictions; (iv) all $\leq$ number restrictions; (v) all predicates over concrete domains.

Concept Contraction. Let $\mathcal{T}$ be an unfoldable TBox in $\mathcal{ALN}$(D) DL, and D and S a demand and supply in $\mathcal{T}$. Then a Concept Contraction Problem between D and S can be solved by Algorithm 4.1. The basic principle is to match separately each of the five concepts sets in the normalized and unfolded expression for D with the corresponding one in S. 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. Hence, a Concept Contraction problem is an extension of a satisfiability one. Since usually one wants to give up as few things as possible, some minimality criteria in the contraction must be defined [17]. In most cases a pure logic-based approach could be not sufficient to decide between which beliefs to give up and which to keep. There is the need of modeling and defining some extra-logical information to be taken into account. For example, we can think to relax some constraints in the original D introducing a *penalty* given by the following function $\prod_C = f\,(p_D, p_S)$ where p_D e p_S are the conflicting predicates respectively within the demand and the supply.

Concept Abduction. A similar structural approach can be pursued to solve a Concept Abduction Problem. Algorithm 4.2 reports the procedure details, where D and S are compared piecewise. Also in this case, one can think to a *penalty function* which increases when the number of missing concepts in S w.r.t. to D grows. The penalty function for the abduction problem can be expressed as $\prod_A = f\,(p_D, p_S)$ where p_D e p_S are missing predicates or restrictions respectively in D and S.

4.2 System Architecture

The mobile reasoning engine was designed to be as modular as possible. Core components provide a framework that supports the two basic system tasks: (1) parsing and manipulation of Knowledge Bases in DIG format and (2) execution of the previously described algorithms for standard and non-standard inference services. Each additional module provides support for a set of logic constructors in both the parsing and reasoning steps. They extend abstract classes and interfaces provided in the core components in order to fit within the software

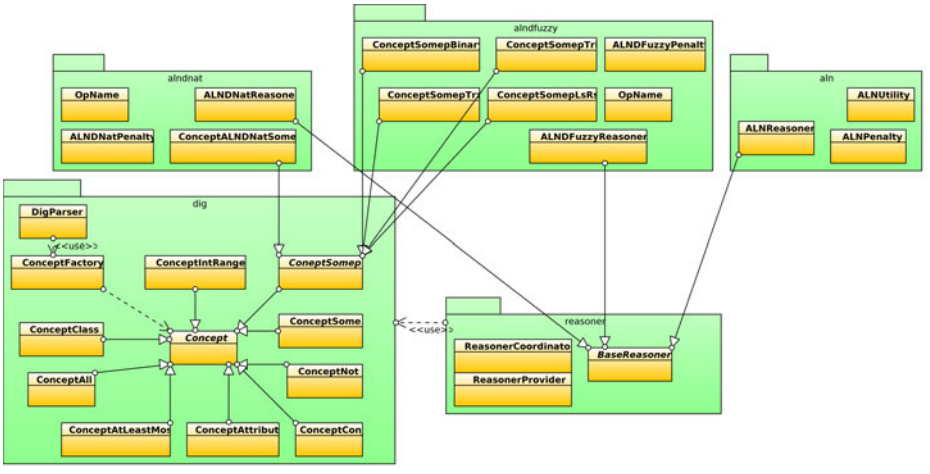

Fig. 1. UML component diagram of the mobile reasoning engine

framework and seamlessly augment the base reasoner capabilities. System architecture is depicted in Figure 1 where solid arrows with full arrowhead indicate inheritance relationship and dashed arrows denote a generic dependency relationship, when not specified further through UML stereotypes (such as the `<<use>>` one). Classes inside each package and their relationships are shown. Hereafter components are individually described in detail.

- **Package** `dig` - Provides the core DIG parsing capabilities, supporting basic $\mathcal{ALN}$(D) constructors. It allows to read a DIG Knowledge Base and build an in-memory model of the TBox. *Factory method* design pattern is used to create Java objects representing concepts during DIG parsing. The abstract class `Concept` provides base fields and methods to represent and process DL concepts. A subclass is defined for each concept type in $\mathcal{ALN}$(D) (atomic concepts, atomic negations, conjunctions, existential and universal quantifications, number restrictions and concrete features).
- **Package** `reasoner` - Implements the backbone of the inference algorithms described in Section 4.1, while allowing them to be extended by additional modules. As explained above, structural inference algorithms –as well as CNF normalization and unfolding– have a strong regularity, where each type of constructors is examined at a time. Therefore, it is possible to extend the support to new constructors by adding blocks within the general algorithmic structure. The same considerations apply to concrete features, which depend on the evaluation of operators over a concrete domain. This scenario is precisely a case for the adoption of the *template method* design pattern. `ReasonerCoordinator` implements the general algorithms for CNF normalization, Unfolding, Satisfiability, Concept Contraction and Concept Abduction. Then, depending on logic constructors and concrete domains used in the KB, it delegates processing of different types of constructs to specialized reasoning modules, which

have been made available by the `ReasonerProvider` class. Each module must extend the abstract class `BaseReasoner`, in order to be integrated in the whole framework and cooperate with classes of this package.

– **Specialized modules** - The remaining packages extend the core ones. Each package P_i provides support for a set S_i of logic constructors and/or concrete domains (with related operators). It is assumed that $S_i \cap S_j = \emptyset \; \forall i, j, i \neq j$, *i.e.*, packages do not overlap. Hence, each constructor can be associated to at most one module and several modules are typically used in an inference procedure. Each P_i must provide: (1) classes extending `dig.Concept` to implement constructors in S_i that are not present in the base `dig` package; (2) a reasoning class (with `-Reasoner` name suffix in Figure 1) that implements parts of inference services involving constructors in S_i; (3) a penalty computation class (with `-Penalty` suffix in Figure 1) that allows to properly compute penalty values related to Concept Contraction and Concept Abduction. Currently implemented modules are:

  - `aln` - Supports $\mathcal{ALN}$ DL. No concrete domains are supported.
  - `alndnat` - Extends $\mathcal{ALN}$ support to $\mathcal{ALN}$(D) DL with the concrete domain of natural numbers and the `<=`, `>=` and `=` operators.
  - `alndfuzzy` - Extends $\mathcal{ALN}$ support to $\mathcal{ALN}$(D) DL with the concrete domain of real numbers; accepted operators are the fuzzy membership functions $trz_{(a,b,c,d)}(\cdot)$ (trapezoidal), $tri_{(a,b,c)}(\cdot)$ (triangular), $ls_{(a,b)}(\cdot)$ (left shoulder) and $rs_{(a,b)}(\cdot)$ (right shoulder) where a, b, c and d are related parameters. Standard DIG syntax was extended in order to express this kind of operators.

Penalty computation requires a deeper examination. The penalty contribution of each conflicting (resp. missing) element e in a Concept Contraction (resp. Abduction) Problem is computed as $p(e) = 1/n$, with n the total number of concepts in the conjunctive expression in which e appears. For example, let us consider the the CNF-normalized formula $A \sqcap B \sqcap \forall R.(C \sqcap D)$. If $e = B$ then $p(e) = 1/3$ because B appears in the conjunction of three elements; however, if $e = \forall R.C$ then $p(e) = 1/2$ because C appears within the conjunction of two concepts. Therefore, total penalty computation is not only additive, but also multiplicative. In order to implement it correctly, during the whole execution of the inference algorithm the `ReasonerCoordinator` manages an in-memory tree containing the elementary penalty values computed by each invocation of a reasoning module. At the end the total penalty can be safely obtained by computing products and sums in the order induced by a visit of the tree.

The devised modular architecture has two main benefits:

1. Efficiency. Only the modules that are needed by constructors used in the current KB are actually invoked.
2. Maintainability. The system can be easily extended to support new logic constructors and concrete domains. Only new modules adhering to the base software framework must be added in order to expand reasoning capabilities to more expressive logic languages or new types of concrete domains with

- $Activity \sqsubseteq Thing$
- $Container \sqsubseteq Thing$
- $Bin \sqsubseteq Container$
- $Cylinder \sqsubseteq Container$
- $FixedTank \sqsubseteq Container$
- $HumanPresence \sqsubseteq Thing$
- $ExternalPresence \sqsubseteq HumanPresence$
- $InternalPresence \sqsubseteq HumanPresence$
- $ExperiencedStaff \sqsubseteq InternalPresence$
- $InexperiencedStaff \sqsubseteq InternalPresence$
- $Substance \sqsubseteq Thing$
- $Fuel \sqsubseteq Substance$
- $CompressedGasFuel \sqsubseteq Fuel$
- $DissolvedGasFuel \sqsubseteq Fuel$
- $LiquefiedGasFuel \sqsubseteq Fuel$
- $LiquefiedPetroleumGas \sqsubseteq LiquefiedGasFuel \sqcap \exists hasFlashPoint \sqcap \forall\, hasFlashPoint(=_{233.15}\, temperature\_K)$
- $LiquidFuel \sqsubseteq Fuel$
- $Petrol \sqsubseteq LiquidFuel \sqcap \exists hasFlashPoint \sqcap \forall\, hasFlashPoint(=_{253.15}\, temperature\_K)$
- $DieselFuel \sqsubseteq LiquidFuel \sqcap \exists hasFlashPoint \sqcap \forall\, hasFlashPoint(=_{338.15}\, temperature\_K)$
- $Combustive \sqsubseteq Substance$
- $CompressedGasCombustive \sqsubseteq Combustive$
- $LiquefiedGasCombustive \sqsubseteq Combustive$
- $RequiredStaffSize \equiv rs_{(4,10)}\, unit$
- $RequiredLiquefiedGasFlashPoint \equiv \exists hasFlashPoint \sqcap \forall\, hasFlashPoint(trz_{(230,273.15,298.15,315)}\, temperature\_K)$
- $RequiredLiquefiedGasContainerMass \equiv \exists hasMass \sqcap \forall\, hasMass(ls_{(75,500)}\, mass\_kg)$

Fig. 2. Relevant axioms in the fire prevention ontology used in the case study

specific operators (*e.g.*, the time domain with Allen's time interval relations or plane geometry with RCC8 relations [18]).

5 Case Study

The proposed framework and the implemented tool have been tested in a fire risk detection case study. The goal was to investigate whether fuzzy $\mathcal{ALN}$(D) was expressive enough to capture the semantics of current Italian regulations[4] about fire prevention certifications.

Basically, fire risk assessment can be modeled as a matchmaking problem. Regulatory requirements play the role of "request" and the description of a location is a "supply". Annotations related to a given context can be conveyed via the semantic-enhanced versions of most common pervasive computing protocols such as Bluetooth, RFID or even ZigBee [2,19]. Semantic descriptions related to a given environment will be automatically extracted from subjects and/or objects dipped into it and put at disposal for reasoning. The more requirements in the request are contradicted or not met by the supply, the highest the penalty will be [14] and the lowest the overall "fire safety score". Concept Contraction and Abduction were used to detect conflicting characteristics and not explicitly fulfilled requirements, respectively.

Both requests and supplies are represented by conjunctive concept expressions referring to the same ontology. A relevant fragment of it is reported in Figure 2, which is necessary to fully understand the following example.

Let us consider a small example extracted from our case study. The example focuses on warehouses of flammable liquefied gas fuel. The concept expressions for the fire safety regulatory requirements and three warehouse instances are reported hereafter.

[4] Italian D.M. 16/02/1982 and later amendments.

D – Requirements for warehouses of flammable liquefied gas fuel in order to receive the safety certification for fire risk prevention in Italy. Only experienced staff must be allowed. Furthermore regulations dictate the required staff size, the maximum quantity of fuel per container and the allowed temperature limits for fuel flash point (the lowest temperature at which it can vaporize to form an ignitable mixture in air). Using concepts appropriately defined in the domain ontology, the fire safety "request" can be expressed as follows:

$$Activity \sqcap \exists hasHumanPresence \sqcap \forall\ hasHumanPresence.(ExperiencedStaff \sqcap RequiredStaffSize \sqcap \neg ExternalPresence) \sqcap \exists containsFuel \sqcap \forall\ containsFuel.(LiquefiedGasFuel \sqcap RequiredLiquefiedGasFlashPoint \sqcap \exists storedIn \sqcap \forall\ storedIn.(Cylinder \sqcap RequiredLiquefiedGasContainerMass))$$

S_1 – This fuel warehouse contains liquefied petroleum gas in 400 kg cylinders. Five expert workers operate in the warehouse, which is also accessible to non-authorized staff members. The description can be expressed in DL w.r.t. the domain ontology as:

$$Activity \sqcap \exists hasHumanPresence \sqcap \forall\ hasHumanPresence.(ExperiencedStaff \sqcap =_5 unit \sqcap ExternalPresence) \sqcap \exists containsFuel \sqcap \forall\ containsFuel.(LiquefiedPetroleumGas \sqcap \exists storedIn \sqcap \forall\ storedIn.(Cylinder \sqcap \exists hasMass \sqcap \forall\ hasMass.(=_{400}\ mass\_kg))$$

S_2 – This fuel warehouse contains liquefied petroleum gas in 200 kg cylinders. Eleven expert workers operate in the warehouse, which is not accessible to other people. The description can be expressed in formula as:

$$Activity \sqcap \exists hasHumanPresence \sqcap \forall\ hasHumanPresence.(ExperiencedStaff \sqcap =_{11} unit \sqcap \neg ExternalPresence) \sqcap \exists containsFuel \sqcap \forall\ containsFuel.(LiquefiedPetroleumGas \sqcap \exists storedIn \sqcap \forall\ storedIn.(Cylinder \sqcap \exists hasMass \sqcap \forall\ hasMass.(=_{200}\ mass\_kg))$$

S_3 – This fuel warehouse contains liquefied petroleum gas in 600 kg cylinders. Three expert workers operate in the warehouse, which is also accessible to non-authorized staff members. The description can be expressed in DL w.r.t. the domain ontology as:

$$Activity \sqcap \exists hasHumanPresence \sqcap \forall\ hasHumanPresence.(ExperiencedStaff \sqcap =_3 unit \sqcap ExternalPresence) \sqcap \exists containsFuel \sqcap \forall\ containsFuel.(LiquefiedPetroleumGas \sqcap \exists storedIn \sqcap \forall\ storedIn.(Cylinder \sqcap \exists hasMass \sqcap \forall\ hasMass.(=_{600}\ mass\_kg))$$

Results are reported in Table 1. The second and third column show the Give Up and Keep part of the result of the Concept Contraction procedure executed by the mobile matchmaker. The last column reports the overall safety score, computed as the percentage of the number of elements in the (unfolded and normalized expression of the) request that are in conflict or missing from the supply, as determined by Concept Contraction and Abduction. Warehouse S_2 is the best matching instance, basically infringing no safety requirement, as can be seen from the Give Up outcome that contains only the top concept. On the contrary, both S_1 and S_3 contradict some requirements, so their score is significantly lower.

At a deeper analysis, it should be noted that S_2 has not a 100% score, even though it has to Give Up no elements. This is due to modeling concrete features –such as number of staff members and substance mass limits– with fuzzy

Table 1. Matchmaking results

Instance	Give Up	Keep	Score
S_1	$G = \forall\ hasHumanPresence.$ $(\neg ExternalPresence)$	$K = Activity \sqcap \exists hasHumanPresence \sqcap$ $\forall\ hasHumanPresence.(ExperiencedStaff \sqcap$ $RequiredStaffSize) \sqcap \exists containsFuel \sqcap$ $\forall\ containsFuel.(LiquefiedGasFuel \sqcap$ $RequiredLiquefiedGasFlashPoint \sqcap$ $\exists storedIn \sqcap \forall\ storedIn(Cylinder \sqcap$ $RequiredLiquefiedGasContainerMass))$	45.6%
S_2	$G = \top$	$K = S_2$	87.2%
S_3	$G = \forall\ hasHumanPresence.$ $(RequiredStaffSize \sqcap$ $\neg ExternalPresence) \sqcap$ $\forall\ containsFuel.(\forall\ storedIn.$ $(RequiredLiquefied$ $GasContainerMass))$	$K = Activity \sqcap \exists hasHumanPresence \sqcap$ $\forall\ hasHumanPresence.ExperiencedStaff \sqcap$ $\exists containsFuel \sqcap$ $\forall\ containsFuel.(LiquefiedGasFuel \sqcap$ $RequiredLiquefiedGasFlashPoint \sqcap$ $\exists storedIn \sqcap \forall\ storedIn.Cylinder)$	33.8%

membership functions, whose truth value is a continuous function in the $[0, 1]$ range. Consequently, a concrete feature in a supply will match semantically the corresponding one in the request *to a certain extent*. Let us consider liquefied gas fuel mass in our example: request prescribes mass to be between 75 kg and 500 kg, with a left-shoulder membership function. This means that containers of 75 kg or less will be "completely safe", those of 500 kg or more will be "completely unsafe" (like in S_3), those in between will have a proportional safety degree. So the match between the requirement in the request and the 200 kg cylinders in S_2 gets a semantic truth degree of $\frac{500-200}{500-75} = 0.706$. Similarly, S_1 has only one element to Give Up entirely but it receives a poor score: this is due to the fact that the number of staff members and fuel cylinder mass are close to the limits specified in the request.

6 Experimental Results

In order to evaluate performance of the proposed system, tests were carried out using the Java ME virtual machine and profiler of Sun Java Wireless Toolkit 2.5.2 for CLDC[5] on a notebook PC with Intel Core Duo T2300 CPU, 1 GB RAM memory and Ubuntu 9.04 operating system[6]. Using a synthetic TBox, 100 request/supply pairs were generated randomly. Generated expressions contain 17.64 ± 4.73 concepts on average (minimum 1, maximum 26) after CNF normalization and unfolding of TBox axioms.

Matchmaking was executed using Concept Contraction and Abduction inference services. Each matchmaking test was executed at two different speed settings for the Java virtual machine, 100 and 1000 bytecode instructions/ms. Based on our experience, these settings are roughly comparable to performance in CPU-bound processes for current mid-range (100 $) Java-equipped mobile phones and high-end (400 $) Java-enabled smartphones, respectively. Execution

[5] Sun Java Wireless Toolkit for CLDC: `http://java.sun.com/products/sjwtoolkit/`

[6] The usage of a simulation environment allowed us to repeat tests with different settings in order to produce more objective results not depending on hardware/software features of a given mobile device so facilitating reproducibility.

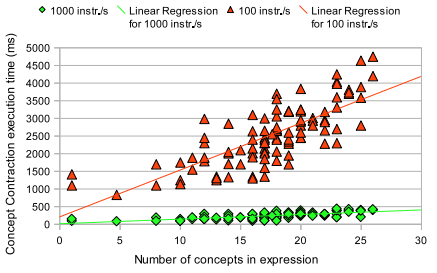

Fig. 3. Concept Contraction performance in 100 tests

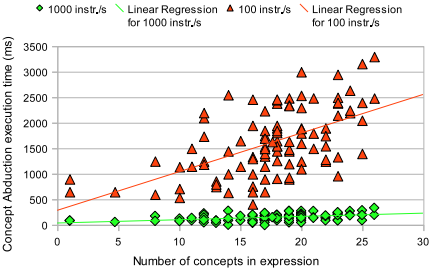

Fig. 4. Concept Abduction performance in 100 tests

time for Contraction and Abduction procedures were evaluated, as well as the peak of main memory allocation above the baseline memory usage, which was measured at idle system state before matchmaking request submission.

Figure 3 and Figure 4 report execution times for Concept Contraction and Abduction, respectively. At 1000 bytecode instructions/ms, computation times are rather low (on average, 248.15 ± 89.71 for Contraction and 161.8 ± 68.74 for Abduction), surely acceptable for mobile scenarios requiring on-the-fly matchmaking. At 100 bytecodes/ms there is a corresponding tenfold increase in turnaround time (2551.59 ± 834.96 for Contraction and 1630.65 ± 648.72 for Abduction on average). From a practical standpoint, using devices with low computational capabilities such as standard mobile phones, latencies would be somewhat high w.r.t. user expectations. It should be noted, however, that time is linear in the size of the input with a good approximation, *e.g.*, at 1000 bytecodes/ms correlation coefficients are $R_c^2 = 0.58$ and $R_a^2 = 0.32$ for Contraction and Abduction respectively. This means that acceptable performance could be often achieved by appropriately limiting query complexity in accordance with the computational capabilities of the particular mobile device.

As a further remark, it can be noticed that Contraction is generally slower than Abduction, but it exhibits a more regular behavior w.r.t. the size of the input. Both observations can be justified by the fact that Abduction process stops as soon as a clash is found between concepts in the request and those in the supply.

Finally, Figure 5 reports the main memory consumption peak during the matchmaking process. Results are obviously independent of virtual machine speed. Obtained values are very low (30.95 ± 9.1 kB on average), evidencing that our modular reasoner architecture does not impair memory efficiency. Memory consumption exhibits a positive correlation with the size of concept expressions (correlation coefficient $R^2 = 0.53$). Consequently, it can be deemed that the matchmaker is capable of dealing with complex $\mathcal{ALN}(\mathrm{D})$ concept expressions even on mid-range Java mobile phones with few hundred kilobytes of memory.

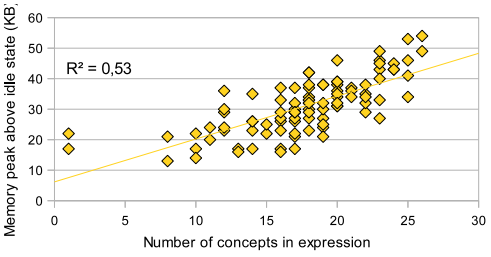

Fig. 5. Main memory usage peak during matchmaking w.r.t. idle state. Linear regression line is plotted.

7 Conclusion and Future Work

We have proposed a novel matchmaking framework useful in contexts where stable networks and powerful workstations are lacking. Original abduction and contraction algorithms presented in [14] have been enriched to support fuzzy DL operators and adapted to handheld devices. The obtained framework has been validated in a fire hazard case study to support in advance emergency detection and recovery. Resulting approach is general purpose as it is fully reusable in several scenarios enabling multiple applications. Early evaluations of system working proved its feasibility and usefulness.

Future work includes: wider tests on the proposed system in order to carry out further optimizations for improving system performances; extension of the prototype to both support more expressive logic formalisms and add resource composition service; finally novel non-standard inferences useful in several application scenarios (such as Least Common Subsumer (LCS) search) will be integrated and tested.

Acknowledgments

The authors acknowledge partial support of Apulia Region Strategic Project PS_121 - Telecommunication Facilities and Wireless Sensor Networks in Emergency Management.

References

1. Di Noia, T., Di Sciascio, E., Donini, F., Mongiello, M.: Semantic matchmaking in a p-2-p electronic marketplace. In: Proceedings of the ACM Symposium on Applied Computing, pp. 582–586 (2003)
2. Di Noia, T., Di Sciascio, E., Donini, F.M., Ruta, M., Scioscia, F., Tinelli, E.: Semantic-based bluetooth-rfid interaction for advanced resource discovery in pervasive contexts. International Journal on Semantic Web and Information Systems 4(1), 50–74 (2008)

3. Baader, F., Calvanese, D., Mc Guinness, D., Nardi, D., Patel-Schneider, P.: The Description Logic Handbook. Cambridge University Press, Cambridge (2002)
4. Koch, F.: 3APL-M platform for deliberative agents in mobile devices. In: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems, p. 154. ACM, New York (2005)
5. Tai, W., Brennan, R., Keeney, J., O'Sullivan, D.: An Automatically Composable OWL Reasoner for Resource Constrained Devices. In: 2009 IEEE International Conference on Semantic Computing, pp. 495–502. IEEE, Los Alamitos (2009)
6. Horrocks, I., Patel-Schneider, P.: Optimizing description logic subsumption. Journal of Logic and Computation 9(3), 267–293 (1999)
7. Sinner, A., Kleemann, T.: KRHyper - In Your Pocket. In: Nieuwenhuis, R. (ed.) CADE 2005. LNCS (LNAI), vol. 3632, pp. 452–457. Springer, Heidelberg (2005)
8. Kleemann, T., Sinner, A.: User Profiles and Matchmaking on Mobile Phones. In: Umeda, M., Wolf, A., Bartenstein, O., Geske, U., Seipel, D., Takata, O. (eds.) INAP 2005. LNCS (LNAI), vol. 4369, pp. 135–147. Springer, Heidelberg (2006)
9. Steller, L., Krishnaswamy, S.: Pervasive Service Discovery: mTableaux Mobile Reasoning. In: International Conference on Semantic Systems (I-Semantics), Graz, Austria (2008)
10. Colucci, S., Di Noia, T., Pinto, A., Ragone, A., Ruta, M., Tinelli, E.: A non-monotonic approach to semantic matchmaking and request refinement in e-marketplaces. International Journal of Electronic Commerce 12(2), 127–154 (2007)
11. Ruta, M., Di Noia, T., Di Sciascio, E., Piscitelli, G., Scioscia, F.: Semantic-based mobile registry for dynamic RFID-based logistics support. In: 10th International Conference on Electronic Commerce. ACM Press, New York (2008) doi: 10.1145/1409540.1409576
12. Ruta, M., Scioscia, F., Di Noia, T., Di Sciascio, E.: Reasoning in Pervasive Environments: an Implementation of Concept Abduction with Mobile OODBMS. In: Proceedings of IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology, pp. 145–148. IEEE, Los Alamitos (2009)
13. Brachman, R., Levesque, H.: The Tractability of Subsumption in Frame-based Description Languages. In: 4th National Conference on Artificial Intelligence (AAAI-84), pp. 34–37. Morgan Kaufmann, San Francisco (1984)
14. Di Noia, T., Di Sciascio, E., Donini, F.: Semantic matchmaking as non-monotonic reasoning: A description logic approach. Journal of Artificial Intelligence Research 29, 269–307 (2007)
15. Bechhofer, S., Möller, R., Crowther, P.: The DIG Description Logic Interface. In: Proceedings of the 16th International Workshop on Description Logics (DL'03), CEUR Workshop Proceedings, Rome, Italy, September 2003, vol. 81 (2003)
16. Straccia, U.: A fuzzy description logic for the semantic web. In: Fuzzy Logic and the Semantic Web, Capturing Intelligence, pp. 167–181. Elsevier, Amsterdam (2005)
17. Gärdenfors, P.: Knowledge in Flux: Modeling the Dynamics of Epistemic States. Bradford Books, MIT Press, Cambridge (1988)
18. Lutz, C., Miličić, M.: A tableau algorithm for description logics with concrete domains and general tboxes. Journal of Automated Reasoning 38(1), 227–259 (2007)
19. Ruta, M., Scioscia, F., Di Noia, T., Di Sciascio, E.: A hybrid ZigBee/Bluetooth approach to mobile semantic grids. International Journal of Computer Systems Science and Engineering (2010)