Match'n'Date: Semantic Matchmaking for Mobile Dating in P2P Environments

Michele Ruta, Tommaso Di Noia, Eugenio Di Sciascio, and Floriano Scioscia

Politecnico di Bari
via Re David 200, I-70125
Bari, ITALY
email: {m.ruta, t.dinoia, disciascio, f.scioscia}@poliba.it

Abstract. In a generic semantic-based matchmaking process, given a request, it is desirable to obtain a ranked list of compatible services/ resources/ profiles in order of relevance. Furthermore, a match explanation can provide useful information to modify or refine the original request in a principled way. Though the feasibility of this approach has been proved with fixed reasoning engines, it is a challenging subject to perform inference tasks on handheld devices. Here we propose abduction and contraction algorithms in Description Logics specifically devised for applications in mobile environments. A simple interaction paradigm based on Bluetooth protocol stack has also been implemented and tested in a mobile *dating* case study.

1 Introduction

We propose a novel discovery framework whose concrete implementation has been carried out in a mobile *dating* case study even if it is cross-applicable in all discovery scenarios. Knowledge Representation techniques and approaches have been shaped to be effectively suitable in volatile ubiquitous computing contexts. In particular, here we adapt abduction and contraction algorithms used in [4] in order to allow their exploitation in resource-constrained contexts. Building on previous work that enhanced the discovery possibilities offered by standard code-based matching procedures with semantic-based capabilities [15], here we devise a further evolution of matchmaking algorithms allowing to run the proposed reasoning services also on mobile devices. This framework and approach has been tested for profile matchmaking in a p2p environment.

Users equipped with a mobile device expose both their semantically annotated profile and preferences they would like to satisfy encountering another user. An exact match between requester preferences and offered profiles is surely the best possible result, but it is probably too rare to be realistic. It is more feasible to obtain a ranked list of available user profiles even if they do not completely fulfill the request. In the same way, when the user preferences and retrieved profiles are incompatible, it could be interesting to know what are the causes for the incongruence if user is willing to retract some constraints she originally

imposed to reach a potential match. The proposed system exploits a revised version of non-monotonic inferences [6] (in particular abduction and contraction) to retrieve compatible profiles arranged in relevance order. A score is computed taking into account the semantic *affinity* between preferences expressed by the user and characteristics found in the available profiles. As explained hereafter, we selected a sublanguage deriving from OWL DL, $\mathcal{AL}(D)$, to model ontologies, preferences and profile annotations whereas the proposed system adopts an enhanced version of DIG 1.1 annotations.

The Bluetooth connectivity of handheld user's device is exploited to allow the data exchange aiming at extending the basic service discovery protocol with semantic capabilities. A "micro-layer" has been integrated within a J2ME[1] application level over the Bluetooth stack in order to enable a simple interchange of semantic annotations between a mobile host performing a query and another one exposing its characteristics. We adopt a simple piconet configuration without stable networked zone servers. Peers are equipped with a Bluetooth interface and they are at the same time able to address requests to other mobile clients as well as to receive and reply to external queries. Each device hosts a semantic facilitator to match on-board user preferences with profiles of users in the neighborhood.

The remaining of the paper is structured as follows: in the next section we motivate the proposed approach and present its background. In Section 3 and Section 4 we move on to the presentation of the theoretical framework. Relevant features of the dating application we implemented are outlined in Section 5 with the aid of a simple illustrative case study. Conclusion closes the paper.

2 Background

Exploiting standard relational databases for resource retrieval, the attributes of the offered and requested resources must exactly coincide to have a match. If requests and offers are simple names or strings, the only possible match would be identity, resulting in an all-or-nothing outcome of the retrieval process. Vague query answering, proposed by [12], was an initial effort to overcome the rigid constraints of relational databases, by attributing weights to several search variables.

Vector-based techniques taken by classical Information Retrieval can be used too, thus reverting the search for a resource matching a request to similarity between weighted vectors of stemmed terms, as proposed in the COINS matchmaker [10] or in LARKS [16]. The need to work in someway with approximation and ranking in DL-based approaches to matchmaking has also recently led to adopting fuzzy-DLs, as in Smart [1] or hybrid approaches, as in the OWLS-MX matchmaker [9].

A further approach structures resource descriptions as set of words. This formalization allows one to evaluate not only identity between sets, but also some

[1] Java 2 Micro Edition: http://java.sun.com/javame/index.jsp

interesting set-based relations between descriptions, such as inclusion, partial overlap, or cardinality of set difference. Anyway, modeling resource descriptions as set of words is too much sensitive to the employed words to be successfully used: the fixed terminology misses meaning that relates to the words. Such a problem can be solved giving to terms a logical and shared meaning through an ontology [8]. Nevertheless set-based approaches already have some properties we believe are fundamental in a resource matchmaking and retrieval process. If we are searching for a resource described through a set of words, we are also interested in sets including the one we search, because they completely fulfill the resource to retrieve. Moreover even if there are characteristics of the retrieved resource not elicited in the description of the searched one, an exact match is still possible because absent information has not to be considered negative. The two statements above may be summarized in the so called *Open World Assumption* (OWA). That is the absence of a characteristic in the description of a resource to be retrieved should not be interpreted as a constraint of absence. Instead it should be considered as a characteristic that could be either refined later or left open if it is irrelevant for the request.

3 Framework and Approach

After discussing the general Knowledge Representation principles that a logical approach to matchmaking may yield, we move on to the Description Logic (DL) setting we adopt[2]. Due to the lack of space, we refer the reader to [6, 4] for several examples and wider argumentation.

3.1 Description Logics and Semantic Matchmaking

From now on we assume that resource descriptions, both requested and offered, in the matchmaking are expressed in a language whose semantics can be mapped to a the Description Logic DL $\mathcal{AL}(D)$, for instance (a subset of) OWL DL or the more compact XML-based DIG language. Such a choice is motivated by several considerations. In [6] it has been proved that there exists a lower bound on the complexity of Concept Contraction, for all DLs that include $\mathcal{AL}$. $\mathcal{AL}(D)$ specifically requires limited computational capabilities to carry out the proposed reasoning services. A simple adaptation of the algorithms reported in the following will allow to report the Concept Contraction and Concept Abduction on an $\mathcal{EL}^{++}$ logic. Formulas (concepts) in $\mathcal{AL}(D)$, we use to represent user profiles and preferences, are built according to the following rules:

$$C, D \rightarrow CN \mid \neg CN \mid \exists R \mid \forall R.C \mid C \sqcap D \mid (\geq_k g) \mid (\leq_k g)$$

where CN represents a concept name. For what concerns the ontology (Terminological Box $\mathcal{T}$ in DL-words) we only allow relations between concept names in the form:

[2] We assume hereafter the reader be familiar with basics of Description Logics formalisms and reasoning [2].

$$\underset{(1)}{CN_1 \sqsubseteq CN_2 \sqcap \ldots CN_n;} \qquad \underset{(2)}{CN_1 \equiv CN_2 \sqcap \ldots CN_n;} \qquad \underset{(3)}{CN_1 \sqsubseteq \neg CN_2 \sqcap \ldots \neg CN_n;}$$

to respectively represent (1) subclass axioms; (2) equivalence axioms; (3) disjoint axioms. Furthermore, given a concept name CN we cannot have more than one equivalence axiom with CN on the left hand side (LHS) and if CN appears on the LHS of an equivalence axiom then it cannot appear on the LHS neither of a subclass axiom nor of a disjoint axiom. In order to avoid cycles within an ontology $\mathcal{T}$, we do not allow a concept name CN appears, directly or indirectly, both on the LHS and on the right hand side of an axiom [2]. Furthermore, for each concrete feature g we impose its range is always explicitely represented by its minimum value and its maximum value. We represent the range of g as:

$$range(g) = (g_{min}, g_{MAX})$$

DL-based systems usually provide two basic reasoning services for $\mathcal{T}$, namely (a) Satisfiability and (b) Subsumption in order to check (a) if a formula C is consistent w.r.t. the ontology $-\mathcal{T} \not\models C \sqsubseteq \bot-$ or (b) if a formula C is more specific or equivalent to a formula D $-\mathcal{T} \models C \sqsubseteq D$.

If we have a *Profile Description* `PD` and a *User Preference* `UP`, we can define at least five different match classes based on subsumption and satisfiability: exact match, subsumption (full) match, plug-in match, intersection (potential) match, disjoint (partial) match [13, 11, 4]. Given a preference, representing a request, and a set of profiles, representing the resources to be retrieved, we can classify the match relation between the preference and each profile according to the above classes. As argued in [4], there is a strong relation among these classes. In particular:

- given a partial match between `UP` and `PD`, solving a Concept Contraction Problem (CCP) [3] one can compute what has to be given up G and kept K in `UP` in order to have a potential match between K (a contracted version of `UP`) and `PD`. Hence, the result of a CCP is a pair $\langle G, K \rangle$ representing respectively elements in `UP` conflicting with `PD` and the (best) contracted `UP` compatible with `PD`.
- given a potential match between `UP` and `PD`, solving a Concept Abduction Problem (CAP) [7] one can compute what has to be hypothesized in `PD` in order to have a full match with `UP` (or its contracted version K). Hence, the result of a CAP is a concept H representing in some way what is *underspecified* in `PD` in order to completely satisfy a preference `UP`. Please note that we say *underspecified* instead of *missing*. This is because we are under a OWA.

Of course, both for Concept Contraction and Concept Abduction we have to define some minimality criteria both on G (give up as few things as possible) and on H (hypothesize as few things as possible). The interested reader may refer to [3, 5] for some minimality criteria in the framework of Description Logics.

An Algorithm for Concept Contraction in $\mathcal{AL}(D)$. An algorithm to solve CAPs for $\mathcal{ALN}$ has been proposed in [6] and it can be easily adapted to deal

with $\mathcal{AL}(D)$. In this section we propose a new algorithm to compute a possible solution to CCPs in $\mathcal{AL}(D)$ given two concepts PD, UP both of them satisfiable w.r.t. an ontology $\mathcal{T}$. Before computing solutions to a CCP it is more convenient, from a computational perspective, to reduce both PD and UP to a common normal form. We use here well know techniques [2] to syntactically transform concepts and preserve their formal semantics with respect to $\mathcal{T}$. Given a concept C the normalization process is performed applying recursively the rewriting rules in Fig.1 to each occurrence of the element appearing in the LHS of the rule.

$$
\begin{array}{rl}
CN_1 \mapsto CN_1 \sqcap CN_2 \sqcap \dots CN_n & \text{if } CN_1 \sqsubseteq CN_2 \sqcap \dots CN_n \in \mathcal{T} \\
CN_1 \mapsto CN_2 \sqcap CN_3 \sqcap \dots CN_n & \text{if } CN_1 \equiv CN_2 \sqcap \dots CN_n \in \mathcal{T} \\
CN_1 \mapsto CN_1 \sqcap \neg CN_2 \sqcap \dots \neg CN_n & \text{if } CN_1 \sqsubseteq \neg CN_2 \sqcap \dots \neg CN_n \in \mathcal{T}
\end{array}
$$

$$
\begin{array}{rl}
C \sqcap \bot \mapsto \bot; & \\
A \sqcap \neg A \mapsto \bot; & \\
(\geq_n g) \mapsto \bot & \text{if } n > g_{MAX}; \\
(\leq_m g) \mapsto \bot & \text{if } m < g_{min}; \\
(\geq_n g) \sqcap (\leq_m g) \mapsto \bot & \text{if } n > m; \\
\forall R.C_1 \sqcap \forall R.C_2 \mapsto \forall R.C_1 \sqcap C_2; & \\
(\geq_n g) \sqcap (\geq_m g) \mapsto (\geq_n R) & \text{if } n > m; \\
(\leq_n g) \sqcap (\leq_m g) \mapsto (\leq_n g) & \text{if } n < m;
\end{array}
$$

Fig. 1. Normalization rules

Note that we refer to acyclic terminologies. In case of cyclic terminologies a simple blocking is enough to guarantee the termination of the normalization process. Given a concept $C \in \mathcal{AL}(D)$ and a taxonomy $\mathcal{T}$, we call $norm(C, \mathcal{T})$ the rewriting of C following the rules in Fig.1. If we consider $norm(C, \mathcal{T})$, it can be always represented as the conjunction $C^{CN} \sqcap C^{R} \sqcap C^{(D)}$, where:

C^{CN} is the conjunction of (negated) concept names;
C^{R} is the conjunction of terms involving roles;
$C^{(D)}$ is the conjunction of concrete domain restrictions, no more than two for every role (the maximum and the minimum for each concrete feature).

With $|norm(C, \mathcal{T})|$ we refer to the length of $norm(C, \mathcal{T})$ computed following Algorithm 1 reported in the follwoing.

At this point we have all the elements we need to formalize an algorithm to solve a CCP in $\mathcal{AL}(D)$ given two concepts PD and UP both satisfiable w.r.t. $\mathcal{T}$. In Algorithm $contract(\mathcal{AL}(D), norm(\texttt{PD}, \mathcal{T}), norm(\texttt{UP}, \mathcal{T}), \mathcal{T})$ starting from the normalized version of UP and PD we compute a solution $\langle G, K \rangle$ to the corresponding CCP and we also return *penalty*: a numerical value representing the worth associated to G. In other words, we compute the cost for a contraction of

Algorithm 1: How to compute the length of a concept C with respect to a taxonomy $\mathcal{T}$

Algorithm: $|norm(C, \mathcal{T})|$
Input: a $\mathcal{AL}(D)$ concept C and a taxonomy $\mathcal{T}$
Output: the length of $norm(C, \mathcal{T})$
$length := 0$;
if $norm(C, \mathcal{T}) = \bot$ **then**
    return 1;
end
foreach *(negated) concept name* $CN \in norm(C, \mathcal{T})^{CN}$ **do**
    $length := length + 1$;
end
foreach $(\geq_n g) \in norm(C, \mathcal{T})^{(D)}$ *or* $(\leq_m g) \in norm(C, \mathcal{T})^{(D)}$ **do**
    $length := length + 1$;
end
foreach $\exists R \in norm(C, \mathcal{T})^R$ **do**
    $length := length + 1$;
end
foreach $\forall R.D$ **do**
    $length := length + |norm(D, \mathcal{T})|$;
end
return *length*;

UP. We will use this value to evaluate the global utility function associated to a profile w.r.t. a set of preferences. Actually, the algorithm can be easily adapted to deal with different penalty functions [6].

Notice that, even though we impose both UP and PD to be satisfiable w.r.t. to $\mathcal{T}$, in lines 1-8 we also consider the case $\mathtt{UP} = \bot$. This is needed because of the recursive nature of the algorithm. In fact, in line 33 we have a recursive call involving the restrictions of a role R. In case this restriction is $\bot$, *i.e.*, $\forall R.\bot$ occurs UP, we have $\mathtt{UP} = \bot$ when we call $contract(\mathcal{AL}(D), norm(\mathtt{PD}, \mathcal{T}), \bot, \mathcal{T})$ in line 33. For the sake of readability of the algorithm let us pose $norm(\mathtt{PD}, \mathcal{T}) = \bar{\mathtt{PD}}$ and $norm(\mathtt{UP}, \mathcal{T}) = \bar{\mathtt{UP}}$.

Algorithm: $contract(\mathcal{AL}(D), \bar{\mathtt{PD}}, \bar{\mathtt{UP}}, \mathcal{T})$

1: $penalty := 0$;
2: **if** $\bar{\mathtt{UP}} = \bot$ **then**
3: **if** $\bar{\mathtt{PD}} \neq \bot$ **then**
4: **return** $(\langle \bot, \top \rangle, 1)$;
5: **else**
6: **return** $(\langle \bot, \top \rangle, 0)$;
7: **end if**
8: **else**
9: $G := \top$;
10: $K := \top \sqcap \bar{\mathtt{UP}}$;

```
11:    if P̄D = ⊥ then
12:      return (⟨UP̄, ⊤⟩, |UP̄|);
13:    end if
14:    for each (negated) concept name CN ∈ K^CN do
15:      for each concept name CN′ ∈ norm(CN, T)^CN do
16:        if there exists CN″ in P̄D^CN such that CN″ = ¬CN′ then
17:          G := G ⊓ CN;
18:          remove CN from K^CN;
19:          penalty := penalty + 1;
20:        end if
21:      end for
22:    end for
23:    for each concept ∃R ∈ K^R do
24:      if there exists ∀R.⊥ ∈ P̄D^R then
25:        G := G ⊓ ∃R;
26:        remove ∃R from K^CN;
27:        penalty := penalty + 1;
28:      end if
29:    end for
30:    for each concept ∀R.E in K^R do
31:      if either there exists ∃R ∈ K^R or there exists ∃R ∈ P̄D^R then
32:        for each concept ∀R.F in P̄D^R do
33:          (⟨G′, K′⟩, penalty′) := contract(AL(D), E, F, T);
34:          G := G ⊓ ∀R.G′;
35:          replace ∀R.E in K with ∀R.K′;
36:          penalty := penalty + penalty′;
37:        end for
38:      end if
39:    end for
40:    for each concept (≥_x g) in K do
41:      if there exists (≤_y g) in P̄D and y < x then
42:        replace (≥_x g) with (≥_y g);
43:        G := G ⊓ (≥_x g);
44:        penalty := penalty + (x−y)/x;
45:      end if
46:    end for
47:    for each concept (≤_x g) in K do
48:      if exists (≥_y g) in P̄D and y > x then
49:        replace (≤_x g) with (≤_y g);
50:        G := G ⊓ (≥_x g);
51:        penalty := penalty + 1 + (y−x)/x;
52:      end if
53:    end for
54:  end if
55:  return (⟨G, K⟩, penalty);
```

4 Dealing with User Preferences

In real dating scenarios it is quite rare to find exactly the profile we are looking for. Often we have to reformulate one or more preferences and to hypothesize some characteristics not specified in the profiles we found. Based on this reformulate/hypothesize process we usually assign a relevance score to the profile representing how good our preferences have been satisfied.

In such a matchmaking process, a user request, can be split often into two separate parts: **strict** requirements and **preferences** [14]. **Strict** requirements represent what, in the request, has to be strictly matched by the retrieved profile description. **Preferences** can be seen as soft user requirements. In other words, the user will accept even a profile whose description does not represent exactly what the user prefers. Usually, a weight is associated to each preference in order to represent its worth (absolute or relative to the other preferences). Hence, for a user preference $\texttt{UP}$ we distinguish between a concept $\texttt{UP}_S$ representing strict requirements and a set of weighted concepts $\langle \texttt{UP}, v \rangle$ where $\texttt{UP}$ is a DL concept and v is a numerical value representing the preference worth. It should be clear that a matchmaking process has not to be performed w.r.t. $\texttt{UP}_S$. It represents what the user is not willing to risk on at all. He does not want to hypothesize nothing on it. An approximate solution would not be significant for $\texttt{UP}_S$. Actually, performing a matchmaking process between preferences and a profile description $\texttt{PD}$ makes more sense. After all, preferences represent what the user *would like* to be satisfied by $\texttt{PD}$. Hence, even though a preference is satisfied *with a certain degree* (not necessarily completely) the user will be satisfied *with a certain degree* as well.

Given an ontology $\mathcal{T}$, a profile description $\texttt{PD}$, a strict requirement $\texttt{UP}_S$ and a set of preferences $\mathcal{P} = \{\langle \texttt{UP}_i, v_i \rangle\}$ we compute a global ranking penalty using Algorithm 2. Here we assign a $penalty \neq +\infty$ to profiles whose description fully satisfies user strict requirements. We also introduce a penalty threshold ϑ. If the global penalty is higher than ϑ then we discard the selected profile setting $penalty := +\infty$ (line 14). Once we have a profile description such that $\mathcal{T} \models \texttt{PD} \sqsubseteq \texttt{UP}_S$, then we compute how much it satisfies user preferences. For each preference we take into account both a numerical evaluation of the characteristics to be given up with $penalty^c$ and a numerical evaluation of those characteristics to be hypothesized in $penalty^a$. The function *abduce* called in line 7 and line 10 is a combination of the algorithms (slightly modified to be used with $\mathcal{AL}(D)$) presented in [6] to compute and rank solutions to CAPs. We do not report here the algorithms for the sake of brevity. In line 12 of Algorithm 2 we combine $penalty^a$ and $penalty^b$ using two parameters h, g representing the worth associated respectively to $penalty^a$ and $penalty^b$.

The value of *penalty* can be easily converted to an affinity value using the following simple transformation:

$$affinity = 1 - \frac{penalty}{|norm(\texttt{UP}, \mathcal{T})|}$$

Algorithm 2: Algorithm for preference-based semantic retrieval

Algorithm: $preference\_retrieve(\texttt{PD}, \texttt{UP}_S, \mathcal{P}, \mathcal{T}, t)$
penalty = 0 ;
if $\mathcal{T} \models \mathit{PD} \sqsubseteq \mathit{UP}_S$ **then**
  foreach $\langle \mathit{UP}_i, v_i \rangle \in \mathcal{P}$ **do**
    if $\mathcal{T} \models \mathit{UP}_i \sqcap \mathit{PD} \sqsubseteq \bot$ **then**
      $(\langle G, K \rangle, penalty^c) := contract(\mathcal{AL}(D), \texttt{PD}, \texttt{UP}_i, \mathcal{T})$;
      $(H, penalty^a) := abduce(\mathcal{AL}(D), \texttt{PD}, K, \mathcal{T})$;
    else
      $penalty^c := 0$;
      $(H, penalty^a) := abduce(\mathcal{AL}(D), \texttt{PD}, \texttt{UP}_i, \mathcal{T})$;
    end
    $penalty := penalty + v_i \cdot (h \cdot penalty^a + g \cdot penalty^c)$;
  end
  if $penalty > t$ **then**
    $penalty := +\infty$;
  end
  return $(penalty, \langle G, K \rangle, H)$;
end
return $(+\infty, \langle \mathit{UP}, \top \rangle, \bot)$;

5 Case Study: `Match'n'Date`

The mobile dating application `Match'n'Date` has been developed from scratch as a case study for the proposed matchmaking framework and algorithms. The goal is to facilitate acquaintance among people in a given environment. The proposed application is a pure peer-to-peer ubiquitous computing tool, based only on Bluetooth wireless ad-hoc networking. The core is a mobile matchmaker implementing reasoning algorithms for Concept Abduction and Concept Contraction. Note that since Concept Abduction extends Subsumption and Concept Contraction extends Satisfiability [6], the reasoner is also able to perform both consistency and subsumption checks. Each user stores her personal *profile* `PD` and a set of preference $\mathcal{P}$ on her device. They refer to a common domain ontology, which models people's physical appearance and personal interests[3].

A typical use case follows the protocol steps reported hereafter (also illustrated in Fig. 2). We refer to the device of the user looking for a profile as α and to the device hosting a discovered profile as β.

1. The user starts `Match'n'Date` on her mobile device (α). It looks for other devices in the Bluetooth radio range.
2. For each found device β, α checks if `Match'n'Date` is currently running and waiting for a connection.

[3] Due to lack of space, the reference ontology is not reported here.

3. If `Match'n'Date` is running on β, then α asks β to send the profile corresponding to her user. So α sends its profile to β. Profile exchange is performed via the Bluetooth OBEX (OBject EXchange) feature[4].
4. Both α and β run Algorithm $preference\_retrieve$ presented in Section 4 and compute their penalty values. β computes $penalty_{\alpha,\beta}$ while α computes $penalty_{\beta,\alpha}$. If $penalty_{\alpha,\beta} = +\infty$, then α sends a `HALT` message to β. Similarly β sends a `HALT` message to α in case $penalty_{\beta,\alpha} = +\infty$. In both cases the interaction between α and β ends.
5. If no `HALT` messages have been sent, then α sends an invitation to β to start a chat session (over Bluetooth).
6. Now β may visualize the profile sent by α. It may check the $affinity_{\alpha,\beta}$ value (see Fig.5) and it may ask for an explanation of the score looking at the values of $\langle G, K\rangle$ and H returned by $preference\_retrieve$.
7. β may accept or decline the invitation from α.

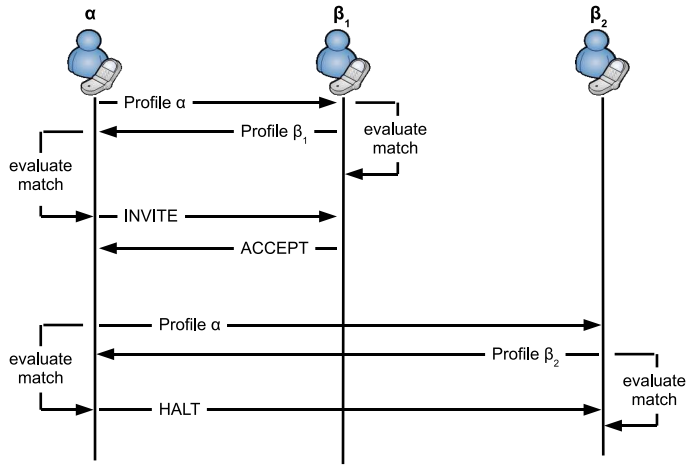

Fig. 2. A typical interaction between `Match'n'Date` devices

5.1 Running Example

Albert has been invited to a party by his room mate Joe, but he is getting quite bored. Joe is spending all the time with his girlfriend and Albert does not know anyone and he cannot find interesting conversation topics with other people. He would like to find a nice and not engaged girl to talk to. After all, Albert does

[4] As the system is at a prototypical state, profiles are now pre-loaded into the handheld. We are developing an intuitive GUI to manage the profile insertion.

Fig. 3. Main application form

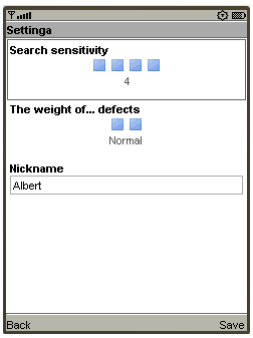

Fig. 4. Settings form

not want to spend all the evening talking with her boyfriend. He would like a woman between 21 and 32 years old and between 160 and 180 cm high, who likes painting and –very important– has not black hair. His former girlfriend had black hair. Currently, he is a little bit biased against black hair girls. So he launches `Match'n'Date` *on his mobile phone. The main menu is shown (as in Fig. 3). Albert selects* `Search` *and* `Match'n'Date` *searches for other compatible devices in its Bluetooth radio range. Fingers crossed.*

A remote device running `Match'n'Date` *is found. It belongs to Barbara, who is getting bored too. The party is full of geeks. The most interesting and hot topics tonight seem to be the very last unstable release of the Linux kernel. Luckily she has* `Match'n'Date` *running on her mobile phone. Albert's device retrieves Barbara's profile and sends his profile to Barbara. The matchmaking process starts.*

Hereafter we report the Albert's preferences in logic formalism. Using the graphical interface presented in Fig.4, Albert is able to set the value of the threshold t and the values for h and g used in line 12 of Algorithm 2. In the current implementation of `Match'n'Date` we use a single parameter and always assume $h = g$.

$\texttt{UP}_S^{Albert}$: $\exists hasMaritalStatus \sqcap \forall hasMaritalStatus.Free$

$\texttt{UP}_1^{Albert}$: $\langle (\geq_{age} 21) \sqcap (\leq_{age} 32) \sqcap (\geq_{height} 160) \sqcap (\leq_{height} 180), 0.3 \rangle$
$\texttt{UP}_2^{Albert}$: $\langle \exists hasHobby \sqcap \forall hasHobby.Painting, 0.2 \rangle$
$\texttt{UP}_3^{Albert}$: $\langle \exists hasHairColor \sqcap \forall hasHairColor.\neg Black, 0.5 \rangle$

Barbara is 28 years old and 172 cm high. She has red hair and currently she is not engaged. She likes art and she does not like swimming. She usually listens to pop-rock music and she watches romantic movies but not science fiction ones.

$\texttt{PD}^{Barbara}$: $(\geq_{age} 28) \sqcap (\leq_{age} 28) \sqcap (\geq_{height} 172) \sqcap (\leq_{height} 172) \sqcap \exists hasHairColor$
$\sqcap \forall hasHairColor.Red \sqcap \exists hasMaritalStatus \sqcap \forall hasMaritalStatus.Free$
$\sqcap \exists hasHobby \sqcap \forall hasHobby.Art$
$\sqcap \exists hasSportPassion \sqcap \forall hasSportPassion.\neg Swimming$

Fig. 5. Matchmaking score form

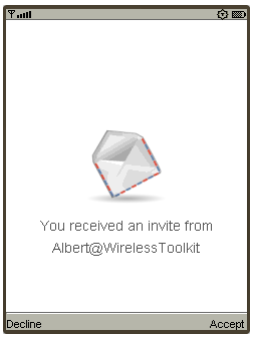

Fig. 6. Invite notification form

$\sqcap \exists favoriteMusicGenre \sqcap \forall favoriteMusicGenre.Pop - Rock$
$\sqcap \exists favoriteMovieGenre \sqcap \forall favoriteMovieGenre.(Romantic \sqcap \neg Sci - Fi)$

Albert is satisfied with the match outcome and wishes to invite Barbara to a chat. The dating application allows the user to contact the remote device for a chat session.

A simple text-based protocol was developed on top of Bluetooth OBEX for this purpose. Upon reception of an invite from α, β displays a notification to Barbara (see Fig. 6), who can either accept or decline the invitation. If β accepts, the chat session starts.

5.2 Experimental Results

One of the main issues in adapting Semantic Web technologies to mobile scenarios is to cope with computational costs. Matchmaking tasks usually need a heavy use of computational resources. This is the most significant reason why we developed our framework limiting the full expressiveness of OWL DL so using its $\mathcal{AL}(D)$ subset. Note that the reasoning algorithms we propose can be executed in polynomial time and they do not need highly optimized data structures.

In what follows we report some performance evaluation tests. In Fig.7, the time (in milliseconds) needed to calculate the affinity value for 100 pairs Preference-Profile randomly generated is shown. The simulation have been conducted exploiting the *Sun Java (TM) Wireless Toolkit 2.5.2 for CLDC*[5] allowing to emulate Virtual Machines (VMs) with different speeds (ranging from 100 to 1000 *bytecode/ms*). In order to cope with limited computational capabilities and reduced memory availability of handhelds, we fixed the speed of VM to 100 bytecode/ms as reference value for our simulations. In the Fig.8, the time (in milliseconds) needed for Concept Contraction –varying the number of concepts and restrictions in each list of preferences– is reported. Finally, Fig.9 shows the time (in milliseconds) needed for Concept Abduction w.r.t. the number of concepts and restrictions in the component *to keep* –K– of each list of preferences.

[5] http://java.sun.com/products/sjwtoolkit/

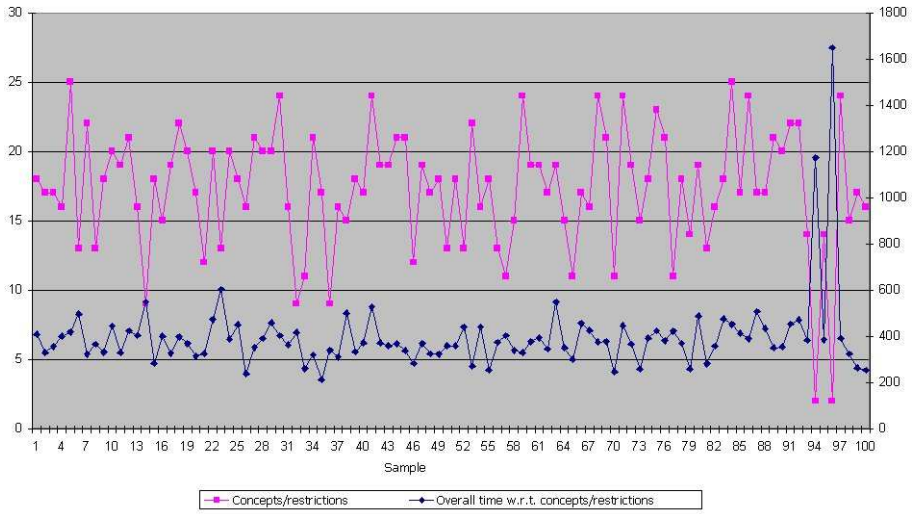

Fig. 7. Overall calculation time w.r.t. concepts and restrictions

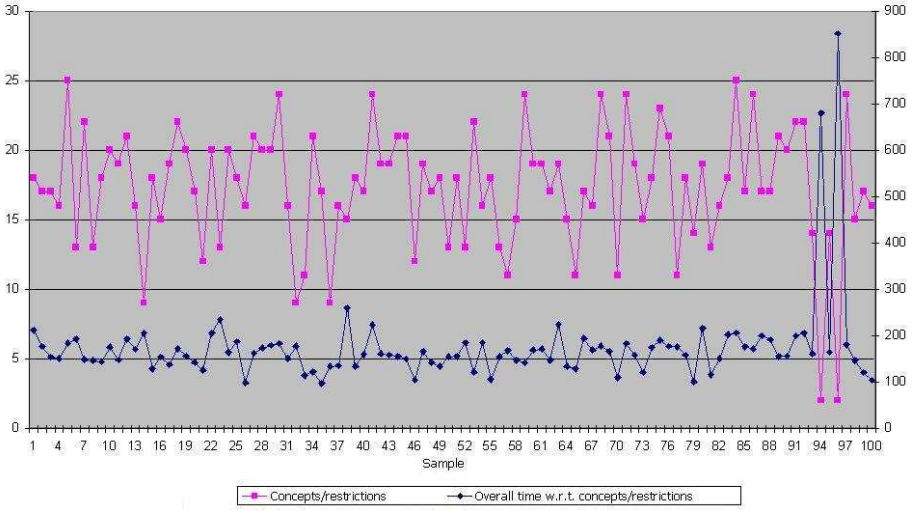

Fig. 8. Execution time for Concept Contraction

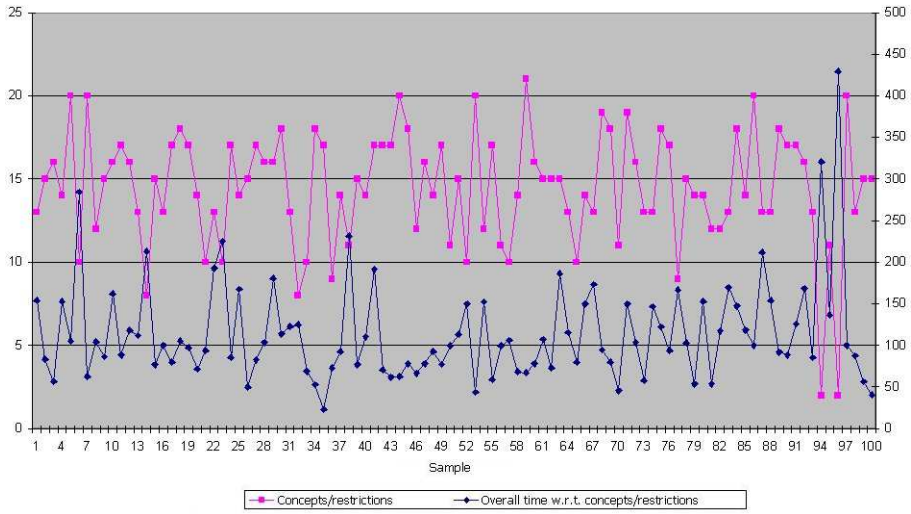

Fig. 9. Execution time for Concept Abduction

6 Conclusion

We have proposed a novel discovery framework for mobile ad-hoc contexts without stable and fixed network infrastructures. Abduction and contraction algorithms presented in [6] have been adapted to allow an exploitation in wireless and p2p scenarios. The proposed approach has been validated in a dating case study where users –equipped with a Bluetooth device– search for semantically annotated profiles compatible with their preferences (also expressed by means of a logic annotation). Framework and approach are general purpose as they are fully re-usable in different contexts and applications.

Future work is aimed at enhancing the expressiveness of the managed logic attempting to remove some constraint actually imposed (as for example the possibility to use the $\exists$ construct for profile definitions). We are currently working on a thorough evaluation of the approach basically measuring the response times of the system in different use cases and with different hardware and network configurations.

Acknowledgments

The authors wish to thank Nicola Caragnano for fruitful discussions and for the implementation of `Match'n'Date`. The authors acknowledge partial support of Apulia Region Strategic Project PS_121 and PS_092.

References

1. S. Agarwal and S. Lamparter. smart - a semantic matchmaking portal for electronic markets. In *Proceedings of the 7th International IEEE Conference on E-Commerce Technology 2005*, 2005.
2. F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002.
3. S. Colucci, T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Concept Abduction and Contraction in Description Logics. In *Proceedings of the 16th International Workshop on Description Logics (DL'03)*, volume 81 of *CEUR Workshop Proceedings*, September 2003.
4. S. Colucci, T. Di Noia, A. Pinto, A. Ragone, M. Ruta, and E. Tinelli. A non-monotonic approach to semantic matchmaking and request refinement in e-marketplaces. *International Journal of Electronic Commerce*, 12(2), 2007.
5. T. Di Noia, E. Di Sciascio, and F.M. 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)*, pages 307–320. 2004.
6. T. Di Noia, E. Di Sciascio, and F.M. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.
7. T. Di Noia, E. Di Sciascio, F.M. Donini, and M. Mongiello. Abductive matchmaking using description logics. In *IJCAI 2003*, pages 337–342, Acapulco, Messico, August 9–15 2003. MK.
8. D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An Ontology Infrastructure for the Semantic Web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.
9. M. Klusch, B. Fries, and K. Sycara. Automated semantic web service discovery with owls-mx. In *In AAMAS 2006*, pages 915–922. ACM Press, 2006.
10. D. Kuokka and L. Harada. Integrating Information Via Matchmaking. 6:261–279, 1996.
11. R. Lara, M.A. Corella, and P. Castells. A flexible model for service discovery on the web. *International Journal of Electronic Commerce – Special Issue on Semantic Matchmaking and Resource Retrieval*, 12(2):11–41, 2007.
12. A. Motro. VAGUE: A User Interface to Relational Databases that Permits Vague Queries. *ACM Transactions on Office Information Systems*, 6(3):187–214, 1988.
13. M. Paolucci, T. Kawamura, T. Payne, and K. Sycara. Semantic matching of web services capabilities. In *Proceedings of the First International Semantic Web Conference (ISWC-02)*, pages 333–347. Springer-Verlag, 2002.
14. A. Ragone, T. Di Noia, E. Di Sciascio, and F.M. Donini. Logic-based automated multi-issue bilateral negotiation in peer-to-peer e-marketplaces. *Autonomous Agents and Multi-Agent Systems Journal*, 16(3):249–270, 2008.
15. M. Ruta, T. Di Noia, E. Di Sciascio, and F.M. Donini. Semantic based collaborative p2p in ubiquitous computing. *Web Intelligence and Agent Systems*, 5(4):375–391, 2007.
16. K. Sycara, S. Widoff, M. Klusch, and J. Lu. LARKS: Dynamic Matchmaking Among Heterogeneus Software Agents in Cyberspace. *Autonomous agents and multi-agent systems*, 5:173–203, 2002.