Answering Fuzzy Preference Queries over Data Web Services

Soumaya Amdouni[1], Mahmoud Barhamgi[1], Djamal Benslimane[1], Allel Hadjali[2], Karim Benouaret[1], and Rim Faiz[3]

[1] LIRIS Laboratory, Claude Bernard Lyon1 University 69622 Villeurbanne, France
[2] Enssat, University of Rennes 1 22305, Lannion, France
[3] University of Carthage-IHEC 2016 Carthage, Tunisia
[4] {samdouni,barhamgi,dbenslim,benouaret}@liris.cnrs.fr, hadjali@enssat.fr, Rim.Faiz@ihec.rnu.tn

Abstract. This paper describes a system that supports preference query answering over a set of data Web services. The proposed system is capable to rank-order the query results in the presence of fuzzy preferences. To do so, we provide different software components organized into two main modules. The first module provides the top-k service compositions. It is mainly based on (*i*) query rewriting techniques to generate relevant services and compositions, (*ii*) fuzzy dominance relationship to rank both individual and composite services. The second module adopts a fuzzy database approach to provide a graded service composition execution engine ranking returned data results.

1 Introduction

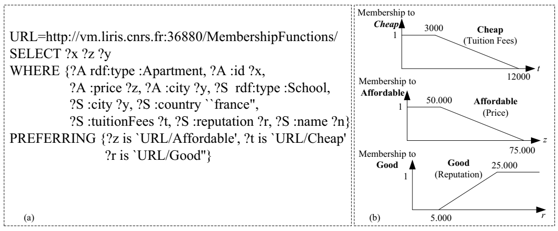
Mashups are situational applications that join data sources to better meet the information needs of Web users. Typically, the access to data sources is carried out through Web services. This type of services is known as Data-as-a-Service [4]. Due to the Web dynamic nature, building mashups at Web scale triggers the need to set up an effective service composition framework that would identify the most relevant services, compose them, and rank the constantly-changing data items accessed by services with respect to user's preferences. In this work, we adopt a flexible approach to model preferences based on fuzzy sets theory [6].

Example. Consider a Web user planning to buy a new apartment. The user would like to find an apartment with an affordable price and located near to

Table 1. Available Web Services

Service	Functionality	Constraints
$S_1(\$c, ?s, ?t, ?r, ?a)$	Returns the schools s along with their tuition fees t, reputation r and addresses a in a given country c	t=*cheap*, r=*high*
$S_2(\$c, ?s, ?t, ?r, ?a)$		t=*expensive*, r=*good*
$S_3(\$a, ?ap, ?p)$	Returns the apartments for sale ap, their prices p at a given address a	p=*affordable*
$S_4(\$a, ?ap, ?p)$		p=*expensive*

high schools with cheap tuition fees and good reputation. A such query Q is described in SPARQL language as in Figure1. Many online data sources (e.g., apartments.com) provide the pricing information of a large set of apartments available for sale. Other yellow-pages provide various information about schools (including their locations, fees, and reputations). Assume that these information are provided by the services in Table1. Input and output parameters are proceeded by "\$" and "?" respectively.



```
URL=http://vm.liris.cnrs.fr:36880/MembershipFunctions/
SELECT ?x ?z ?y
WHERE {?A rdf:type :Apartment, ?A :id ?x,
         ?A :price ?z, ?A :city ?y, ?S  rdf:type :School,
         ?S :city ?y, ?S :country ``france",
         ?S :tuitionFees ?t, ?S :reputation ?r, ?S :name ?n}
PREFERRING {?z is `URL/Affordable', ?t is `URL/Cheap'
              ?r is `URL/Good"}
```

Fig. 1. (a)SPARQL query Q, (b) the associated membership functions

Challenges. Answering the fuzzy query Q over data services raises the following two main challenges:(1) *Computing the best service compositions answering the fuzzy preference queries.* This challenge necessitates to understand the semantics of the published services, to retain the most relevant services that better satisfy the user's preferences and to generate the best k compositions that satisfy the query. (2) *Ranking the results of fuzzy preference queries.* Data returned from the services invocations may partially satisfy the fuzzy preferences of the query. It is then important for a given service composition to rank-order results it may return to express how good results they are.

2 System Overview

Our system is composed of the following two modules:

2.1 Top-K Web Service Compositions

The Top-k service compositions module is provided to compute the best k compositions that answer the user query. The processing of this component is shown in window (a) Figure 2. We briefly describe below its different components, more details are provided in [1,2].

Query Rewriting. The component *RDF Query Rewriter* is provided to identify the relevant services that match all or parts of the user query. It exploits the semantic descriptions of services given in the form of SPARQL queries.

Fuzzy Constraints Matching. The *Fuzzy Constraint Matcher* component is used to compute the matching degrees between the fuzzy preference constraints of the query and the fuzzy service constraints for each relevant service. Four distinct Fuzzy Constraints Matching Methods are used and implemented to associate to each relevant service 4 degrees. Such degrees express to what extent a fuzzy service constraint matches a fuzzy query constraint.

Services Ranking. Our proposed *Services Ranker* component uses a fuzzy dominance to express the extent to which a matching degree dominates another one and associates fuzzy score with individual service.

In this step, we propose a fuzzy dominating score (FDS) for individual services. An FDS of a service S indicates the average extent to which S dominates a set of services, those answering the same subquery. Moreover, it associates fuzzy score with a composition. The score of a composition is computed as an aggregation of the scores of its services.

Top-k Compositions. This component is provided to efficiently generate the compositions that better answer a fuzzy preference query. Instead of generating all possible compositions, we compute their scores and return the top-k compositions, we provide an optimization technique that eliminates some relevant services for which we are sure that if they are composed with other ones, the obtained compositions are not in the top-k.

2.2 Query Results Ranking

The results returned by a composition may be large which may cause the users to miss the ones that are most relevant to their prefrences. We propose a fuzzy database approach to rank data returned by service composition execution [3]. Each relation R obtained from a service invocation is extended to include a new column noted *grade* that expresses to what degree a tuple t of R satisfies the fuzzy predicates and graded relation is noted by R^g. The graded relations are orchestrated using a graded relational algebra.

Formally speaking, assuming a fuzzy predicates set $FP = P_1 \wedge P_2 \wedge ... \wedge P_d$, where P_i, $i = 1...d$, is a fuzzy predicate (such that x is "cheap"...) and $\wedge$ stands for the conjunction connector. Window (b) in Figure 2 shows how the user can edit and test different fuzzy terms. A service composition execution plan is displayed on window (c). Generated service composition execution plan is expressed in terms of graded relational algebraic operators which are an adaptation of relational algebraic operators to the graded relations. The following set of graded operators are defined.

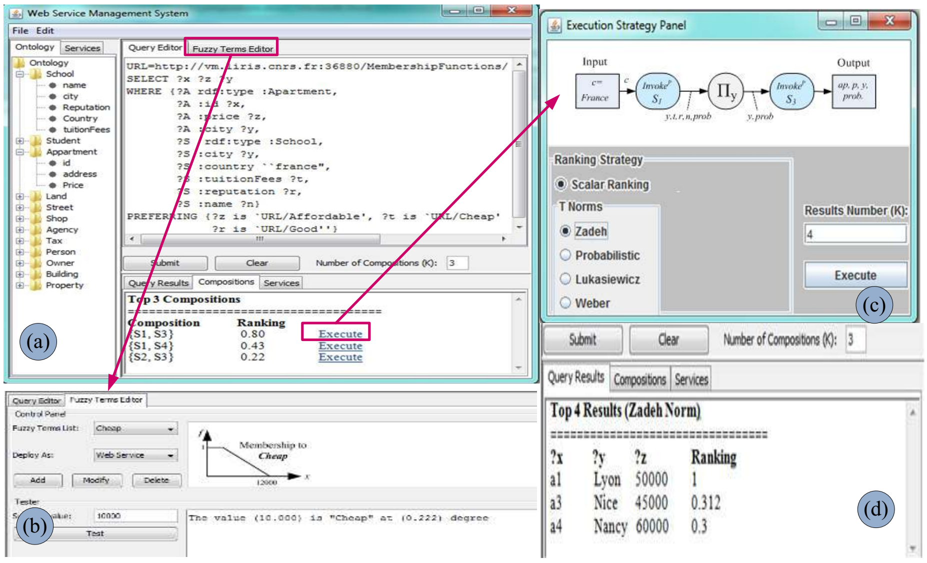
- *The Graded Invocation Invoke*$^g(S, t^g_{in}, O^g)$: Let S be a service, t^g_{in} the graded input tuple with which S is invoked, O^g the graded output, and $S.O$ be the output of S. The *Invoke*^{g} computes $g_1(t_i) = \top(\mu_{P_1(t_i)}, \mu_{P_2(t_i)}, ..., \mu_{P_n(t_i)})$ where $\top$ is a t-norm operator and μ_{P_i} the membership function associated with P_i. Our system implements the following t-norms: *Zadeh*, *Probabilistic*, and *Lukasiewicz.*
- *Graded Join*: $\infty^g(I^g_1, I^g_2)$, where I^g_1 and I^g_2 are two graded data sets. The grade of an outputted tuple is given by: $g(\infty^g(t, t')) = \top(g(t), g(t'))$ where $\top$ is a t-norm, and t and t' are tuples from I^g_1 and I^g_2 respectively.

– *Graded Projection* $\prod_{A}^{g}$. The projection is an operation that selects specified attributes A=$\{a_1, a_2, ...\}$ from a results set. The grade of an outputted tuple t is: $g(t) = \perp (g(t'_1), .., g(t'_i), .., g(t'_n))$ where $t = \prod_A (t'_i)_{i=1:n}$ and $\perp$ is the co-norm corresponding to the t-norm $\top$ used in the graded join.
– *Graded Union* $\cup^g$. The grade of an outputted tuple t is: $g(t) = \perp (g(t'_1), .., g(t'_i), .., g(t'_n))$, where $t'_i = t$ and $i = 1 : n$

The *Ranking aware Execution Engine* implements the defined operators and the final ranked results are displayed in window (d).

3 Demo Highlights

The demo will show all of the components in figure 2. To illustrate the robustness of our approach in different settings, we apply our scenario on a set of 200 different data services, accessing a synthetic dataset containing information about a consequent data objects of the real-estate application domain.



```
URL=http://vm.liris.cnrs.fr:36880/MembershipFunctions/
SELECT ?x ?z ?y
WHERE {?A rdf:type :Apartment,
        ?A :id ?x,
        ?A :price ?z,
        ?A :city ?y,
        ?S  rdf:type :School,
        ?S :city ?y,
        ?S :country ``france",
        ?S :tuitionFees ?t,
        ?S :reputation ?r,
        ?S :name ?n}
PREFERRING {?z is `URL/Affordable', ?t is `URL/Cheap'
            ?r is `URL/Good''}
```

Number of Compositions (K): 3

```
Top 3 Compositions
=====================================
Composition       Ranking
{S1, S3}           0.80        Execute
{S1, S4}           0.43        Execute
{S2, S3}           0.22        Execute
```

Fuzzy Terms List: Cheap

Deploy As: Web Service

Sample value: 10000

```
The value (10.000) is "Cheap" at (0.222) degree
```

[✓ Scalar Ranking]

T Norms: [✓ Zadeh] Probabilistic Lukasiewicz Weber

Results Number (K): 4

Number of Compositions (K): 3

```
Top 4 Results (Zadeh Norm)
==================================
?x    ?y     ?z      Ranking
a1    Lyon   50000   1
a3    Nice   45000   0.312
a4    Nancy  60000   0.3
```

Fig. 2. Demo - Compositions and Results Ranking

References

1. Barhamgi, M., Benslimane, D., Medjahed, B.: A query rewriting approach for web service composition. IEEE T. Services Computing 3(3), 206–222 (2010)
2. Benouaret, K., Benslimane, D., HadjAli, A., Barhamgi, M.: Fudocs: A web service composition system based on fuzzy dominance for preference query answering. PVLDB 4(11), 1430–1433 (2011)

3. Bosc, P., Buckles, B.B., Petry, F.E., Pivert, O.: Fuzzy Databases, vol. 3, pp. 403–468. Kluwer Academic Publishers (1999)
4. Carey, M.J.: Soa what? IEEE Computer 41(3), 92–94 (2008)
5. Dubois, D., Prade, H.: Beyond min agregation in multicriteria decision: (ordered) weighted mean, discri-min, leximin, pp. 181–192. K.A.P (1997)
6. Zadeh, L.A.: Fuzzy sets. Information and Control 8(3), 338–353 (1965)