Semantic Reasoning for Web Services Discovery

Boualem Benatallah†, Mohand-Said Hacid‡,
Christophe Rey§ and Farouk Toumani§

† SCSE, UNSW, Sydney, Australia (boualem@cse.unsw.edu.au)
‡ LIRIS, UCB Lyon I, France (mshacid@liris.univ-lyon1.fr)
§ LIMOS, UBP, France ({rey, ftoumani}@isima.fr)
Outline

- The context
- Semantic service discovery
- The best profile covering problem
- Implementation and experimentation
- Related work
- Conclusion
The context: Semantic Web Services

- Motivation
  take the Web technologies a step further by providing foundations to enable **automated** discovery, access, combination, and management of Web services

- Two main research issues
  - Providing rich and machine understandable representation of services properties, capabilities, and behavior
  - Providing reasoning mechanisms to support automation activities

  → **Focus on service discovery**
Semantic service discovery

“The beauty of the e-services vision is the ability to find the currently available service that best fits my needs” [Casati01]

• Discovering services based on their capabilities
  Semantic comparison between a service request and available services

• Study in the context of DAML-S
  – An ontology for describing web services
  – Based on DAML+OIL
    can be regarded as an expressive description logic
DAML-S service profile

- Describes the service capabilities
  Functional representation (among others) in terms of Inputs/Outputs
- Used for advertising and discovering services
  - Service advertisements
  - Service requests
The proposed approach

- Comparing requests with services based on their inputs and outputs

- A novel matching algorithm
  - Service discovery as a rewriting process
    - a service request $\rightsquigarrow$ the closest subset of services
  - Compute the extra information:
    - * Required by a service request but not provided by any existing service
    - * Required by the selected services but not provided by the request

- Formal framework based on description logics
## Illustrating example

<table>
<thead>
<tr>
<th>Service</th>
<th>Inputs</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToTravel</td>
<td>Itinerary, Arrival</td>
<td>TripReservation</td>
</tr>
<tr>
<td>FromTravel</td>
<td>Itinerary, Departure</td>
<td>TripReservation</td>
</tr>
<tr>
<td>Hotel</td>
<td>Destination, StayDuration</td>
<td>HotelReservation</td>
</tr>
</tbody>
</table>

- **ToTravel** allowing to reserve a trip given an itinerary and the arrival time and date
- **FromTravel** allowing to reserve a trip given an itinerary and the departure time and date
- **Hotel** allowing to reserve a hotel given a destination place, a check-in date and a check-out date
Example of a tourism ontology

\[
\begin{align*}
\text{Itinerary} & \equiv (\geq 1 \text{departurePlace}) \sqcap (\forall \text{departurePlace.Location}) \sqcap \\
& \quad (\geq 1 \text{arrivalPlace}) \sqcap (\forall \text{arrivalPlace.Location}) \\
\text{Arrival} & \equiv (\geq 1 \text{arrivalDate}) \sqcap (\forall \text{arrivalDate.Date}) \sqcap \\
& \quad (\geq 1 \text{arrivalTime}) \sqcap (\forall \text{arrivalTime.Time}) \\
\text{Departure} & \equiv (\geq 1 \text{departureDate}) \sqcap (\forall \text{departureDate.Date}) \sqcap \\
& \quad (\geq 1 \text{departureTime}) \sqcap (\forall \text{departureTime.Time}) \\
\text{Destination} & \equiv (\geq 1 \text{destinationPlace}) \sqcap (\forall \text{destinationPlace.Location}) \\
\text{StayDuration} & \equiv (\geq 1 \text{checkIn}) \sqcap (\forall \text{checkIn.Date}) \sqcap \\
& \quad (\geq 1 \text{checkOut}) \sqcap (\forall \text{checkOut.Date}) \\
\text{TripReservation} & \equiv \ldots \\
\text{HotelReservation} & \equiv \ldots \\
\text{CarRental} & \equiv \ldots
\end{align*}
\]
Example of a service request

Q: looks for a vacation package that combines a trip with a hotel and a car rental, given a departure place, an arrival place, a departure date a (hotel) destination place and the check-in and check-out dates.

We write

\[ I(Q) \equiv (\geq 1 \text{departurePlace}) \sqcap (\forall \text{departurePlace}.\text{Location}) \sqcap (\geq 1 \text{arrivalPlace}) \sqcap (\forall \text{arrivalPlace}.\text{Location}) \sqcap (\geq 1 \text{departureDate}) \sqcap (\forall \text{departureDate}.\text{Date}) \sqcap (\geq 1 \text{destinationPlace}) \sqcap (\forall \text{destinationPlace}.\text{Location}) \sqcap (\geq 1 \text{checkIn}) \sqcap (\forall \text{checkIn}.\text{Date}) \sqcap (\geq 1 \text{checkOut}) \sqcap (\forall \text{checkOut}.\text{Date}) \]

\[ O(Q) \equiv \text{TripReservation} \sqcap \text{HotelReservation} \sqcap \text{CarRental} \]
Example of a matching

Consider the following two solutions:

• Solution 1: FromTravel, Hotel
  – Generated outputs: TripReservation, HotelReservation
  – Missed outputs: CarRental
  – Missed inputs: departureTime

• Solution 2: ToTravel, Hotel
  – Generated outputs: TripReservation, HotelReservation
  – Missed outputs: CarRental
  – Missed inputs: arrivalTime, arrivalDate
Statement of the problem

Given a service request $Q$ and a DAML-S ontology $\mathcal{T}$, compute the best combination $E$ of Web services such that:

- $E$ satisfies \textit{as much as possible} the outputs of the request $Q$
- $E$ requires \textit{as little as possible} of inputs that are not provided in the description of $Q$

$E$ is called a \textit{best profile cover} of $Q$ using $\mathcal{T}$
A difference operator

Teege’s Definition [Teege94]

Let $C, D$ be two concept descriptions with $C \subseteq D$

$$C - D := \max \{ B \mid B \cap D \equiv C \}$$

**Remark** the difference is not always semantically unique

- Example

  $C \equiv (\forall R. \bot)$

  $D \equiv (\forall R.P) \cap (\forall R.P')$

  The following two concepts $B_1 \equiv (\forall R.\neg P)$ and $B_2 \equiv (\forall R.\neg P')$ are both members of the set $C - D$. 
Characterizing the description language

- Structural subsumption characterizes the languages where the difference operation is always semantically unique [Teege94]

- Example of such logics: the description logic $\mathcal{L}_1$
  - $\cap, \cup, \top, \bot, (\geq n R), (\exists R.C), (\exists f.C)$ for concepts,
  - bottom ($\bot$), composition ($\circ$), differentiation ($|$) for roles,
  - bottom ($\bot$) and composition ($\circ$) for features

- We consider restricted DAML-S ontologies built using a subset of DAML+OIL for which a structural subsumption algorithm exists
Let $\mathcal{T}$ be a restricted DAML-S ontology, $E$ be a conjunction of some services occurring in $\mathcal{T}$ and $Q$ a service request

- **Profile cover of $Q$ using $\mathcal{T}$:**
  \[ O(Q) - O(E) \neq O(Q) \]

- **Profile rest: outputs of $Q$ not generated by $E$**
  \[ Prest_E(Q) \equiv O(Q) - O(E) \]

- **Profile miss: inputs of $E$ not provided by $Q$**
  \[ Pmiss_E(Q) \equiv I(E) - I(Q) \]
The best profile covering problem

- Best profile cover
  - $E$ is a Pcover of $Q$ using $T$, and
  - there doesn’t exist a Pcover $E'$ of $Q$ using $T$ such that
    $$(|P_{rest}^{E'}(Q)|, |P_{miss}^{E'}(Q)|) < (|P_{rest}^E(Q)|, |P_{miss}^E(Q)|),$$
    where $<$ stands for the lexicographic order.

- The best profile covering problem
  compute all the best profile covers of $Q$ using $T$

- The best profile covering problem is NP-Hard
Computing best profile covers

$(T, Q) \leftrightarrow$ a weighted hypergraph $\mathcal{H}_{TQ}$

- The web services become vertices in $\mathcal{H}_{TQ}$
- Each vertex in $\mathcal{H}_{TQ}$ is associated with a cost equal to the $Pmiss$ of the corresponding service
- The outputs of (a normal form) of $Q$ become edges in $\mathcal{H}_{TQ}$

Computing best profile covers of $Q$ using $T \Leftrightarrow$
Finding the minimal transversals with a minimal cost of $\mathcal{H}_{TQ}$
A Service discovery algorithm

computeBProfileCov: an algorithm for computing the best profile covers

- Based on hypergraph theory
- Makes an improvement over the classical approach (e.g., [Gottlob91, Mannila92]) for computing the minimal transversals
- Implemented as a Java prototype
  - 6 versions of the computeBProfileCov algorithm (different combinations of optimization options)
  - a tool that enables to generate random XML-based services ontologies and associated service requests
Experiments

- Validation in an e-commerce area on small ontologies
- Evaluation of the performance of the algorithm on synthetic ontologies
  - A theoretical study of complexity to characterize the worst cases w.r.t. the number of transversals and the number of elementary operations of the algorithm
  - Experiments on three configurations
  - Performed on a PC with a Pentium III 500 MHz and 384 Mo of RAM
## First results

<table>
<thead>
<tr>
<th>Configurations</th>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of defined concepts in the application domain ontology</td>
<td>365</td>
<td>1334</td>
<td>3405</td>
</tr>
<tr>
<td>Number of web services</td>
<td>366</td>
<td>660</td>
<td>570</td>
</tr>
<tr>
<td>Number of (atomic) clauses in the query</td>
<td>6</td>
<td>33</td>
<td>12</td>
</tr>
<tr>
<td>Overall time results</td>
<td>&lt; 2 secs</td>
<td>&lt; 30 secs</td>
<td>&lt; 2 secs</td>
</tr>
</tbody>
</table>
Related work

- Semantic service discovery
- Query (concept) rewriting
  Intensively investigated in the Database area
Semantic service discovery

- Several matching techniques
  Process Query Language [Bernstein02], Inference rules [Chakraborty01], Syntactic, operational and semantic similarities [Cardoso2002], subsumption and consistency tests [Castillo01], semantic distance between concepts in the ontology [Paolucci02, Payne01]

- Similar approach to [Payne01, Cardoso02, Paolucci02], but a different matching algorithm
  - A global reasoning mechanism
  - A flexible matching process that goes beyond subsumption tests
  - Effective computation of the missed information
Relation with query rewriting

A general framework for rewriting using terminologies [Baader00a]:

- given a terminology $\mathcal{T}$, a concept description $Q$ that does not contain concept names defined in $\mathcal{T}$ and a binary relation $\rho$ between concept descriptions, can $Q$ be rewritten into a description $E$, built using (some) of the names defined in $\mathcal{T}$, such that $Q \rho E$?

- some optimality criterion is defined in order to select the relevant rewritings
Relation with query rewriting (cont.)

Already investigated instances of the general framework:

- Rewriting queries using views (cf. [Halevy2002] for a survey)
  - Maximally-contained rewritings
    \( \rho \) is instanciated by subsumption and the optimality criterion is the inverse subsumption
  - Equivalent rewriting
    \( \rho \) is instanciated by equivalence and the optimality criterion is the cost of the corresponding query plan

- Minimal rewriting problem [Baader00a]
  \( \rho \) is instantiated by equivalence and the optimality criterion is the size of the rewriting
Relation with query rewriting (cont.)

- Best profile covering problem
  A new instance of the general rewriting framework
  - $\rho$ corresponds to concept cover
  - optimality criterion: the lexicographic order of $(|P \text{rest}|, |P \text{miss}|)$
Conclusion and on-going work

- Generic approach: can be applied to other service ontologies than DAML-S
- Extension to languages where the difference operation is not semantically unique
  - $\mathcal{ALN}$: good trade-off between expressivity and complexity
  - Definition of a restricted difference operation to avoid meaningless decompositions of the bottom ($\bot$) concept
  - Formalization of the best covering problem in the presence of inconsistencies
  - An hypergraph-based approach is still valid (but need non-trivial extensions)
- Can service composition be viewed as a kind of query rewriting?
Thanks

more technical details:
http://www.isima.fr/limos/publications.htm