On automating Web services discovery

Boualem Benatallah, Mohand-Said Hacid, Alain Leger, Christophe Rey, Farouk Toumani

VLDB Journal (2005) 14: 84–96 / Digital Object Identifier (DOI) 10.1007/s00778-003-0117-x

On automating Web services discovery

Boualem Benatallah[1], **Mohand-Said Hacid**[2], **Alain Leger**[3], **Christophe Rey**[4], **Farouk Toumani**[4]

[1] School of Computer Science and Engineering, University of New South Wales, Sydney, Australia (e-mail: boualem@cse.unsw.edu.au)
[2] LIRIS, Université Lyon I, France (e-mail: mshacid@liris.univ-lyon1.fr)
[3] France Telecom R&D (e-mail: alain.leger@rd.francetelecom.com)
[4] LIMOS, Université Blaise Pascal, France (e-mail: {rey, ftoumani}@isima.fr)

Edited by V. Atluri. Received: December 15, 2002 / Accepted: September 15, 2003
Published online: February 6, 2004 –

Abstract. One of the challenging problems that Web service technology faces is the ability to effectively discover services based on their capabilities. We present an approach to tackling this problem in the context of description logics (DLs). We formalize service discovery as a new instance of the problem of rewriting concepts using terminologies. We call this new instance the *best covering problem*. We provide a formalization of the *best covering problem* in the framework of DL-based ontologies and propose a hypergraph-based algorithm to effectively compute best covers of a given request. We propose a novel matchmaking algorithm that takes as input a service request (or query) Q and an ontology $\mathcal{T}$ of services and finds a set of services called a "best cover" of Q whose descriptions contain as much *common information* with Q as possible and as little *extra information* with respect to Q as possible. We have implemented the proposed discovery technique and used the developed prototype in the context of the *Multilingual Knowledge Based European Electronic Marketplace* (MKBEEM) project.

Keywords: Web services Discovery – Semantic matchmaking – Description logics – Hypergraphs

1 Introduction

Semantic Web services are emerging as a promising technology for the effective automation of services discovery, combination, and management [17,18,28]. They aim at leveraging two major trends in Web technologies, namely, *Web services* and the *Semantic Web*:

- Web services built upon XML as a vehicle for exchanging messages across applications. The basic technological infrastructure for Web services is structured around three major standards: SOAP (Simple Object Access Protocol), WSDL (Web Services Description Language), and UDDI (Universal Description, Discovery, and Integration) [9,35]. These standards provide the building blocks for service description, discovery, and communication. While Web services technologies have clearly influenced positively the potential of the Web infrastructure by providing programmatic access to information and services, they are hindered by lack of rich and machine-processable abstractions to describe service properties, capabilities, and behavior. As a result of these limitations, very little automation support can be provided to facilitate effective discovery, combination, and management of services. Automation support is considered the cornerstone of effective and efficient access to services in large, heterogeneous, and dynamic environments [9,10,17]. Indeed, until recently the basic Web services infrastructure was used mainly to build simple Web services such as those providing information search capabilities to an open audience (e.g., stock quotes, search engine queries, auction monitoring).
- The Semantic Web aims at improving the technology to organize, search, integrate, and evolve Web-accessible resources (e.g., Web documents, data) by using rich and machine-understandable abstractions for the representation of resources semantics. Ontologies are proposed as means to address semantic heterogeneity among Web-accessible information sources and services. They are used to provide metadata for the effective manipulation of available information including discovering information sources and reasoning about their capabilities. Efforts in this area include the development of ontology languages such as RDF, DAML, and DAML+OIL [14]. In the context of Web services, ontologies promise to take interoperability a step further by providing rich description and modeling of services properties, capabilities, and behavior.

By leveraging efforts in both Web services and the Semantic Web, the Semantic Web services paradigm promises to take Web technologies a step further by providing foundations to enable automated discovery, access, combination, and management of Web services. Efforts in this area focus on providing rich and machine-understandable representation of services properties, capabilities, and behavior as well as reasoning mechanisms to support automation activities [8, 11,13,18,17,28]. Examples of such efforts include DAML-S [13], WSMF (Web Services Modeling Framework) [18], and METEOR-S (http://lsdis.cs.uga.edu/proj/meteor/SWP.htm). Work in this area is still in its infancy. Many of the objectives of the Semantic Web services paradigm, such as description of service capabilities, dynamic service discovery, and goal-driven composition of Web services, have yet to be reached.

Our work focuses on the issue of dynamic discovery of Web services based on their capabilities. Dynamic service discovery is usually based on the rationale that services are selected, at run-time, based on their properties and capabilities. Our aim is to ground the discovery process on a matchmaking between a requester query and available Web service descriptions. We formalize the service discovery approach in the context of description logics (DLs) [15]. A key aspect of DLs is their formal semantics and reasoning support. DLs provide an effective reasoning paradigm for defining and understanding the structure and semantics of concept description ontologies. This is essential for providing formal foundations for the envisioned Semantic Web paradigm [24,26,25]. Indeed, DLs have heavily influenced the development of some Semantic Web ontology languages (e.g., DAML+OIL or OWL [36]). Our work aims at enhancing the potential of Web services by focusing on formal foundations and flexible aspects of their discovery. More specifically, we make the following contributions:

- **Flexible matchmaking between service descriptions and requests.** We propose a matchmaking technique that goes beyond simple subsumption comparisons between a service request and service advertisements. As emphasized in [31], a service discovery mechanism should support flexible matchmaking since it is unrealistic to expect service requests and service advertisements to match exactly. To cope with this requirement, we propose to use a *difference operation* on service descriptions. Such an operation enables one to extract from a subset of Web service descriptions the part that is semantically common with a given service request and the part that is semantically different from the request. Knowing the former and the latter allows one to effectively select relevant Web services. We propose a novel matchmaking algorithm that takes as input a service request (or query) Q and an ontology $\mathcal{T}$ of services and finds a set of services called a "best cover" of Q whose descriptions contain as much *common information* with Q as possible and as little *extra information* with respect to Q as possible.
- **Concept rewriting for effective service matchmaking.** We formalize service matchmaking as a new instance of the problem of rewriting concepts using terminologies [3, 23]. We call this new instance the *best covering problem*. We provide a formalization of the *best covering problem* in the context of DL-based ontologies and propose a hypergraph-based algorithm to effectively compute best covers of a given request.
- **Characterization of service discovery automation in DAML-S service ontologies.** We investigate the reasoning problem associated with service discovery in DAML-S ontologies and its relationship with the expressiveness of the language used to express service descriptions. To study the feasibility of our approach, we have implemented the proposed discovery technique and used the developed prototype in the context of *Multilingual Knowledge Based European Electronic Marketplace* (MKBEEM) project.[1]

[1] *http://www.mkbeem.com*

Organization of the paper

The remainder of this paper is organized as follows. Section 2 provides an overview of the basic concepts of description logics. Section 3 describes the formalization of service discovery in the context of DL-based ontologies. Section 4 presents the hypergraph-based algorithm for computing best covers. An extension of our approach to accommodate DAML-S ontologies is presented in Sect. 5. Section 6 describes an implementation of the proposed service discovery technique and discusses some preliminary experimental results. We review related work in Sect. 7 and provide concluding remarks in Sect. 8.

2 Description logics: an overview

Our approach uses description logics (DLs) [1] as a formal framework. DLs are a family of logics that were developed for modeling complex hierarchical structures and for providing a specialized reasoning engine to perform inferences on these structures. The main reasoning mechanisms (e.g., subsumption or satisfiability) are decidable for the main DLs [15]. Recently, DLs have heavily influenced the development of the Semantic Web languages. For example, DAML+OIL, the ontology language used by DAML-S, is in fact an alternative syntax for a very expressive DL [26].

In this section, we first give basic definitions, and then we describe the notion of difference between descriptions that is the core operation used in our framework.

2.1 Basic definitions

Description logics allow one to represent a domain of interest in terms of *concepts or descriptions* (unary predicates) that characterize subsets of the objects (*individuals*) in the domain and *roles* (binary predicates) over such a domain. Concepts are denoted by expressions formed by means of special constructs. Examples of DL constructs considered in this paper are:

- The symbol $\top$ is a concept description that denotes the top concept, while the symbol $\perp$ stands for the bottom concept.
- Concept conjunction ($\sqcap$), e.g., the concept description $parent \sqcap male$, denotes the set of fathers (i.e., male parents).
- The universal role quantification ($\forall R.C$), e.g., the description $\forall child.male$, denotes the set of individuals whose children are all male.
- The number restriction constructs ($\geq n\ R$) and ($\leq n\ R$), e.g., the description ($\geq 1\ child$), denotes the set of parents (i.e., individuals having at least one child), while the description ($\leq 1\ Leader$) denotes the set of individuals that cannot have more than one leader.

The various DLs differ from one another in the set of constructs they allow. Table 1 shows the constructs of two DLs: $\mathcal{FL}_0$ and $\mathcal{ALN}$. A concept obtained using the constructs of a DL $\mathcal{L}$ is called an $\mathcal{L}$-concept. The semantics of a concept description is defined in terms of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$,

Table 1. Syntax and semantics of some concept-forming constructs

Construct name	Syntax	Semantics	$\mathcal{FL}_0$	$\mathcal{ALN}$
Concept name	P	$P^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$	X	X
Top	$\top$	$\Delta^{\mathcal{I}}$	X	X
Bottom	$\bot$	$\emptyset$		X
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	X	X
Primitive negation	$\neg P$	$\Delta^{\mathcal{I}} \setminus P^{\mathcal{I}}$		X
Universal quantification	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$	X	X
At least number restriction	$(\geq\ nR), n \in \mathbb{N}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}}\} \geq n\}$		X
At most number restriction	$(\leq\ nR), n \in \mathbb{N}$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}}\} \leq n\}$		X

which consists of a nonempty set $\Delta^{\mathcal{I}}$, the domain of the interpretation, and an interpretation function $\cdot^{\mathcal{I}}$, which associates with each concept name $P \in \mathcal{C}$ a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and with each role name $R \in \mathcal{R}$ a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. Additionally, the extension of $.^{\mathcal{I}}$ to arbitrary concept descriptions is defined inductively, as shown in the third column of Table 1. Based on this semantics, subsumption, equivalence, and the notion of a least common subsumer (lcs) are defined as follows.[2] Let $C, C_1, \ldots, C_n$ and D be concept descriptions:

- C is *subsumed by* D (denoted by $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.
- C is *equivalent to* D (denoted $C \equiv D$) iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.
- D is a least common subsumer of $C_1, \ldots, C_n$ (denoted by $D = lcs(C_1, \ldots, C_n)$) iff:
  (1) $C_i \sqsubseteq D$ for all $1 \leq i \leq n$ and
  (2) D is the least concept description with this property, i.e., if D' is a concept description satisfying $C_i \sqsubseteq D'$ for all $1 \leq i \leq n$, then $D \sqsubseteq D'$ [2].

The *intentional* descriptions contained in a knowledge base built using a DL is called *terminology*. The kind of terminologies we consider in this paper are defined below.

Definition 1 ***(terminology)***
Let A be a concept name and C be a concept description. Then $A \doteq C$ is a concept definition. A terminology $\mathcal{T}$ is a finite set of concept definitions such that each concept name occurs at most once on the left-hand side of a definition.

A concept name A is called a defined concept in the terminology $\mathcal{T}$ iff it occurs on the left-hand side of a concept definition in $\mathcal{T}$. Otherwise, A is called an atomic concept.

An interpretation $\mathcal{I}$ satisfies the statement $A \doteq C$ iff $A^{\mathcal{I}} = C^{\mathcal{I}}$. An interpretation $\mathcal{I}$ is a *model* for a terminology $\mathcal{T}$ if $\mathcal{I}$ satisfies all the statements in $\mathcal{T}$.

A terminology built using the constructs of a language $\mathcal{L}$ is called an $\mathcal{L}$-terminology.[3] Below we assume that a terminology $\mathcal{T}$ is *acyclic*, i.e., cyclic dependencies between concept definitions do not exist. Acyclic terminologies can be unfolded by replacing defined names by their definitions until no more defined names occur on the right-hand sides. Therefore, the notion of least common subsumer (lcs) of a set of descriptions can be straightforwardly extended to concepts containing defined names. In this case, we use the expression $lcs_{\mathcal{T}}(C, D)$ to denote the least common subsumer of the concepts C and D with respect to a terminology $\mathcal{T}$ (i.e., the lcs is applied to the unfolded descriptions of C and D).

2.2 The difference operation

In this section, we recall the main results obtained by Teege in [34] regarding the difference operation between two concept descriptions.

Definition 2 ***(Difference operation)***
Let C, D be two concept descriptions with $C \sqsubseteq D$. The difference $C - D$ of C and D is defined by $C - D := \max_{\sqsupseteq}\{B \mid B \sqcap D \equiv C\}$.

The difference of two descriptions C and D is defined as a description containing all information that is part of the description C but not part of the description D. This definition of difference operation requires that the second operand subsumes the first one. However, if the operands C and D are incomparable with respect to the subsumption relation, then the difference $C - D$ can be given by constructing the least common subsumer of C and D, that is, $C - D := C - lcs(C, D)$.

It is worth noting that, in some DLs, the set $C - D$ may contain descriptions that are not semantically equivalent, as illustrated by the following example.

Example 1 Consider the descriptions $C \doteq (\forall R.P_1) \sqcap (\forall R.\neg P_1)$ and $D \doteq (\forall R.P_2) \sqcap (\forall R.(\leq\ 4S))$. The set $C - D$ includes, among others, the nonequivalent descriptions $(\forall R.\neg P_2)$ and $(\forall R.(\geq 5S))$.

Teege [34] provides sufficient conditions to characterize the logics where the difference operation is always semantically unique and can be syntactically realized by constructing the set difference of subterms in a conjunction. Some basic notions and important results of this work are introduced below.

Definition 3 ***(Reduced clause form and structure equivalence)***
Let $\mathcal{L}$ be a description logic.

- *A* ***clause*** *in $\mathcal{L}$ is a description A with the following property: $(A \equiv B \sqcap A') \Rightarrow (B \equiv \top) \vee (B \equiv A)$. Every conjunction $A_1 \sqcap \ldots \sqcap A_n$ of clauses can be represented by the clause set $\{A_1, \ldots, A_n\}$.*

[2] Informally, a least common subsumer of a set of concepts corresponds to the most specific description that subsumes all the given concepts [2].

[3] Henceforth we use the terms terminology and ontology interchangeably.

- $A = \{A_1, \ldots, A_n\}$ *is called a* ***reduced*** *clause set if either* $n = 1$ *or no clause subsumes the conjunction of the other clauses:* $\forall 1 \le i \le n : A_i \not\sqsupseteq A \setminus A_i$. *The set* A *is then called a* ***reduced clause form*** *(RCF) of every description* $B \equiv A_1 \sqcap \ldots \sqcap A_n$.
- *Let* $A = \{A_1, \ldots, A_n\}$ *and* $B = \{B_1, \ldots, B_m\}$ *be reduced clause sets in a description logic* $\mathcal{L}$. *A and B are* ***structure equivalent*** *(denoted by* $A \cong B$*) iff:* $n = m \wedge \forall 1 \le i \le n \, \exists 1 \le j, k \le n : A_i \equiv B_j \wedge B_i \equiv A_k$.
- *If in a description logic for every description all its RCFs are structure equivalent, we say that RCFs are* ***structurally unique*** *in that logic.*

The *structural difference operation* is defined as the set difference between clause sets where clauses are compared on the basis of the equivalence relationship.

Let us now introduce the notion of *structural subsumption* as defined in [34].

Definition 4 ***(structural subsumption)***
The subsumption relation in a description logic $\mathcal{L}$ *is said to be* ***structural*** *iff for any clause* $A \in \mathcal{L}$ *and any description* $B = B_1 \sqcap \ldots \sqcap B_m \in \mathcal{L}$ *that is given by its RCF, the following holds:* $A \sqsupseteq B \Leftrightarrow \exists 1 \le i \le m : A \sqsupseteq B_i$

Teege [34] provides two interesting results: (1) in DLs with structurally unique RCFs, the difference operation can be straightforwardly determined using the *structural difference operation*; and (2) *structural subsumption* is a sufficient condition for a DL to have structurally unique RCFs. Consequently, structural subsumption is a sufficient condition that allows one to identify logics where the difference operation is semantically unique and can be implemented using the *structural difference operation*. However, it is worth noting that the definition of structural subsumption given in [34] is different from the one usually used in the literature. Unfortunately, a consequence of this remark is that many DLs for which a structural subsumption algorithm exists (e.g., $\mathcal{ALN}$ [30]) do not have structurally unique RCFs. Nevertheless, the result given in [34] is still interesting in practice since there exists several DLs that satisfy this property. Examples of such logics include the language $\mathcal{FL}_0 \cup (\ge n\ R)$, which we have used in the context of the MKBEEM project, or the more powerful description logic $\mathcal{L}_1$ [34], which contains the following constructs:

- $\sqcap, \sqcup, \top, \perp, (\ge\ n\ R)$, existential role quantification $(\exists R.C)$ and existential feature quantification $(\exists f.C)$ for concepts, where C denotes a concept, R a role, and f a feature (i.e., a functional role);
- Bottom ($\perp$), composition ($\circ$), differentiation ($|$) for roles;
- Bottom ($\perp$) and composition ($\circ$) for features.

In the remainder of this paper we use the term *structural subsumption* in the sense of [34].

Size of a description. Let $\mathcal{L}$ be a DL with structural subsumption. We define the size $|C|$ of an $\mathcal{L}$-concept description C as the number of clauses in its RCFs.[4] If necessary, a more precise measure of a size of a description can be defined by taking into account the size of each clause (e.g., by counting the number of occurrences of concept and role names in each clause). However, in this case one must use some kind of canonical form to deal with the problem of different descriptions of equivalent clauses. It should be noted that in a DL with structurally unique RCFs, it is often possible to define a canonical form that is itself an RCF [34].

[4] Recall that, since $\mathcal{L}$ has structurally unique RCFs, all the RCFs of an $\mathcal{L}$-description are equivalent and thus have the same number of clauses.

3 Formalization of the best covering problem

In this section, we first formalize the *best covering problem* in the framework of DLs with structural subsumption. Then we describe how to compute best covers using a hypergraph-based algorithm.

3.1 Problem statement

Let us first introduce some basic definitions that are required to formally define the *best covering problem.* Let $\mathcal{L}$ be a DL with structural subsumption, $\mathcal{T}$ an $\mathcal{L}$-terminology, and $Q \not\equiv \perp$ a coherent $\mathcal{L}$-concept description. The set of defined concepts occurring in $\mathcal{T}$ is denoted as $\mathcal{S}_{\mathcal{T}} = \{S_i, i \in [1, n]\}$ with $S_i \not\equiv \perp, \forall i \in [1, n]$. Below we assume that concept descriptions $S_i, i \in [1, n]$ are represented by their RCFs.

Definition 5 ***(Cover)***
A cover of Q *using* $\mathcal{T}$ *is a conjunction* E *of some names* S_i *from* $\mathcal{T}$ *such that:* $Q - lcs_{\mathcal{T}}(Q, E) \not\equiv Q$.

Hence, a cover of a concept Q using $\mathcal{T}$ is defined as any conjunction of concepts occurring in $\mathcal{T}$ that shares some common information with Q. It is worth noting that a cover E of Q is always consistent with Q (i.e., $Q \sqcap E \not\equiv \perp$) since $\mathcal{L}$ is a DL with structurally unique RCFs and $Q \not\equiv \perp$ and $S_i \not\equiv \perp, \forall i \in [1, n]$.[5]

To define the notion of *best cover*, we need to characterize the part of the description of a cover E that is not contained in the description of the query Q and the part of the query Q that is not contained in the description of its cover E.

Definition 6 ***(Rest and miss)***
Let Q *be an* $\mathcal{L}$*-concept description and* E *a cover of* Q *using* $\mathcal{T}$. *The rest of* Q *with respect to* E*, denoted by* $Rest_E(Q)$*, is defined as follows:* $Rest_E(Q) \equiv Q - lcs_{\mathcal{T}}(Q, E)$.
The missing information of Q *with respect to* E*, denoted by* $Miss_E(Q)$*, is defined as follows:* $Miss_E(Q) \equiv E - lcs_{\mathcal{T}}(Q, E)$.

Now we can define the notion of *best cover*.

Definition 7 ***(Best cover)***
A concept description E *is called a best cover of* Q *using a terminology* $\mathcal{T}$ *iff*

- E *is a cover of* Q *using* $\mathcal{T}$*and*

[5] If the language $\mathcal{L}$ contains the incoherent concept $\perp$, then $\perp$ must be a clause, i.e., nontrivial decompositions of $\perp$ are not possible (which means we cannot have incoherent conjunctions of coherent clauses); otherwise it is easy to show that $\mathcal{L}$ does not have structurally unique RCFs.

- *there exists no cover E' of Q using $\mathcal{T}$ such that $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$, where $<$ is the lexicographic order operator.*

A best cover is defined as a cover that has, first, the smallest rest and, second, the smallest miss.

The *best covering problem*, denoted by $\mathcal{BCOV}(\mathcal{T}, Q)$, is then the problem of computing all the best covers of Q using $\mathcal{T}$.

Theorem 1 ***(Complexity of $\mathcal{BCOV}(\mathcal{T}, Q)$)*** *The best covering problem is NP-hard.*

The proof of this theorem follows from the results regarding the minimal rewriting problem [3] (see [22] for a detailed proof).

3.2 Mapping best covers to hypergraph transversals

Let us first recall some necessary definitions regarding hypergraphs.

Definition 8 ***(Hypergraph and transversals)*** *[16]*
A hypergraph $\mathcal{H}$ is a pair (Σ, Γ) of a finite set $\Sigma = \{V_1, \ldots, V_n\}$ and a set Γ of subsets of Σ. The elements of Σ are called vertices, and the elements of Γ are called edges. A set $T \subseteq \Sigma$ is a transversal of $\mathcal{H}$ if, for each $\varepsilon \in \Gamma$, $T \cap \varepsilon \neq \emptyset$. A transversal T is minimal if no proper subset T' of T is a transversal. The set of the minimal transversals of a hypergraph $\mathcal{H}$ is denoted as $Tr(\mathcal{H})$.

In this section, we describe how to express the best covering problem as the problem of finding the minimal transversals with a minimal cost of a given hypergraph.

Definition 9 ***(Hypergraph generation)***
Let $\mathcal{L}$ be a DL with structural subsumption, $\mathcal{T}$ an $\mathcal{L}$-terminology, and Q an $\mathcal{L}$-concept description. Given an instance $\mathcal{BCOV}(\mathcal{T}, Q)$ of the best covering problem, we build a hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ as follows:

- *Each concept name S_i in $\mathcal{T}$ is associated with a vertex V_{S_i} in the hypergraph $\mathcal{H}_{\mathcal{T}Q}$. Thus $\Sigma = \{V_{S_i}, i \in [1, n]\}$.*
- *Each clause $A_i \in Q$, for $i \in [1, k]$, is associated with an edge in $\mathcal{H}_{\mathcal{T}Q}$, denoted by w_{A_i}, with $w_{A_i} = \{V_{S_i} \mid S_i \in \mathcal{S}_{\mathcal{T}}$ and $A_i \in_{\equiv} lcs_{\mathcal{T}}(Q, S_i)\}$, where $\in_{\equiv}$ stands for the membership test modulo equivalence of clauses and $lcs_{\mathcal{T}}(Q, S_i)$ is given by its RCF.*

Notation For the sake of clarity we introduce the following notation. For any set of vertices $X = \{V_{S_i}\}$, subset of Σ, we use the expression $E_X \equiv \sqcap_{V_{S_i} \in X} S_i$ to denote the concept obtained from the conjunction of concept names corresponding to the vertices in X. Conversely, for any concept $E \equiv \sqcap_{j \in [1,m]} S_{i_j}$, we use the expression $X_E = \{V_{S_{i_j}}, j \in [1, m]\}$ to denote the set of vertices corresponding to the concept names in E.

Lemmas 1 and 2, given below, say that computing a cover of Q using $\mathcal{T}$ that minimizes the *rest* amounts to computing a transversal of $\mathcal{H}_{\mathcal{T}Q}$ by considering only the nonempty edges. Proofs of these lemmas are presented in [22].

Lemma 1 ***(Characterization of the minimal rest)***
Let $\mathcal{L}$ be a DL with structural subsumption, $\mathcal{T}$ an $\mathcal{L}$-terminology, and Q an $\mathcal{L}$-concept description. Let $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ be the hypergraph built from the terminology $\mathcal{T}$ and the concept $Q = A_1 \sqcap \ldots \sqcap A_k$ provided by its RCF. The minimal rest (i.e., the rest whose size is minimal) of rewriting Q using $\mathcal{T}$ is: $Rest_{min} \equiv A_{j_1} \sqcap \ldots \sqcap A_{j_l}, \forall j_i \in [1, k] | w_{A_{j_i}} = \emptyset$.

From the previous lemma we know that the minimal rest of rewriting a query Q using $\mathcal{T}$ is always unique and equivalent to $Rest_{min}$.

Lemma 2 ***(Characterization of covers that minimize the rest)***
Let $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$ be the hypergraph built by removing the empty edges from $\mathcal{H}_{\mathcal{T}Q}$. A rewriting $E \equiv S_{i_1} \sqcap \ldots \sqcap S_{i_m}$, with $1 \leq m \leq n$ and $S_{i_j} \in \mathcal{S}_{\mathcal{T}}$ for $1 \leq j \leq m$, is a cover of Q using $\mathcal{T}$ that minimizes the rest iff $X_E = \{V_{S_{i_j}}, j \in [1, m]\}$ is a transversal of $\widehat{\mathcal{H}}_{\mathcal{T}Q}$.

This lemma characterizes the covers that minimize the rest. Consequently, computing the best covers will consist of determining, from those covers, the ones that minimize the miss. To express miss minimization in the hypergraphs framework, we introduce the following notion of cost.

Definition 10 ***(Cost of a set of vertices)***
Let $\mathcal{BCOV}(\mathcal{T}, Q)$ be an instance of the best covering problem and $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$ its associated hypergraph. The cost of the set of vertices X is defined as follows: $cost(X) = |Miss_{E_X}(Q)|$.

Therefore, the best covering problem can be reduced to the computation of the transversals with minimal cost of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Clearly, it is only interesting to consider minimal transversals. In a nutshell, the $\mathcal{BCOV}(\mathcal{T}, Q)$ problem can be reduced to the computation of the minimal transversals with minimal cost of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Therefore, we can reuse and adapt known techniques for computing minimal transversals (e.g., see [7,16,27]) for solving the best covering problem.

3.3 Illustrating example

To illustrate the best covering problem, let us consider a terminology $\mathcal{T}$ containing the following concepts (Web services):

- $ToTravel$: allows one to search for trips given a departure place, an arrival place, an arrival date, and an arrival time.
- $FromTravel$: allows one to search for trips given a departure place, an arrival place, a departure date, and a departure time.
- $Hotel$: allows one to search for hotels given a destination place, a check-in date, a check-out date, the number of adults, and the number of children.

The terminology $\mathcal{T}$, depicted in Table 2, is described using the description logic $\mathcal{FL}_0 \cup \{\geq n\, R\}$.[6]

[6] We denote by $\mathcal{FL}_0 \cup (\geq n\, R)$ the description logic $\mathcal{FL}_0$ augmented with the construct $(\geq n\, R)$.

Table 2. Example of a terminology

ToTravel	$\equiv$	(≥ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location) $\sqcap$ ($\geq$ 1 arrivalDate) $\sqcap$ ($\forall$ arrivalDate.Date) $\sqcap$ ($\geq$ 1 arrivalTime) $\sqcap$ ($\forall$ arrivalTime.Time)
FromTravel	$\equiv$	($\geq$ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location) $\sqcap$ ($\geq$ 1 departureDate) $\sqcap$ ($\forall$ departureDate.Date) $\sqcap$ ($\geq$ 1 departureTime) $\sqcap$ ($\forall$ departureTime.Time)
Hotel	$\equiv$	Accommodation $\sqcap$ ($\geq$ 1 destinationPlace) $\sqcap$ ($\forall$ destinationPlace.Location) $\sqcap$ ($\geq$ 1 checkIn) $\sqcap$ ($\forall$ checkIn.Date) $\sqcap$ ($\geq$ 1 checkOut) $\sqcap$ ($\forall$ checkOut.Date) $\sqcap$ ($\geq$ 1 nbAdults) $\sqcap$ ($\forall$ nbAdults.Integer) $\sqcap$ ($\geq$ 1 nbChildren) $\sqcap$ ($\forall$ nbChildren.Integer)

Let us consider now the following query description:

$Q \equiv$ ($\geq$ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location) $\sqcap$ ($\geq$ 1 departureDate) $\sqcap$ ($\forall$ departureDate.Date) $\sqcap$ Accommodation $\sqcap$ ($\geq$ 1 destinationPlace) $\sqcap$ ($\forall$ destinationPlace.Location) $\sqcap$ ($\geq$ 1 checkIn) $\sqcap$ ($\forall$ checkIn.Date) $\sqcap$ ($\geq$ 1 checkOut) $\sqcap$ ($\forall$ checkOut.Date) $\sqcap$ carRental

We assume that the concept names (e.g., Location, Date, Accommodation) that appear in the description of the query Q and/or in the concept descriptions of $\mathcal{T}$ are all atomic concepts. Hence, the query Q and the concepts of $\mathcal{T}$ are all provided by their RCFs.[7] Therefore, the associated hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ consists of the set of vertices $\Sigma = \{V_{ToTravel}, V_{FromTravel}, V_{Hotel}\}$ and the set of edges:

$\Gamma = \{w_{(\geq 1 departurePlace)}, w_{(\forall departurePlace.Location)}, w_{(\geq 1 arrivalPlace)}, w_{(\forall arrivalPlace.Location)}, w_{(\geq 1 departureDate)}, w_{(\forall departureDate.Date)}, w_{Accommodation}, w_{(\geq 1 destinationPlace)}, w_{(\forall destinationPlace.Location)}, w_{(\geq 1 checkIn)}, w_{(\forall checkIn.Date)}, w_{(\geq 1 checkOut)}, w_{(\forall checkOut.Date)}, w_{carRental}\}$.

The hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ is depicted in Fig. 1. We can see that no concept covers the clause corresponding to the edge $w_{carRental}$ (as we have $w_{carRental} = \emptyset$). Since this is the only empty edge in Γ, the best covers of Q using $\mathcal{T}$ will have exactly the following rest: $Rest_{min} \equiv carRental$. Now, considering the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$, the only minimal transversal is: $X = \{V_{FromTravel}, V_{Hotel}\}$. Thus $E_X \equiv Hotel \sqcap FromTravel$ is the best cover of Q using the terminology $\mathcal{T}$. The size of the missing information of E_X is obtained from the transversal X as shown below:

$cost(X) = |Miss_{FromTravel \sqcap Hotel}(Q)|$ = |($\geq$ 1 departureTime) $\sqcap$ ($\forall$ departureTime.Time) $\sqcap$ ($\geq$ 1 nbAdults) $\sqcap$ ($\forall$ nbAdults.Integer) $\sqcap$ ($\geq$ 1 nbChildren) $\sqcap$ ($\forall$ nbChildren.Integer)| = 6.

In this example, we do not consider this cost because the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ has only one minimal transversal.

[7] Otherwise, we have to recursively unfold the concept (resp. query) description by replacing by its definition each concept name appearing in the concept (resp. query) description.

4 Computing best covers

In this section we present an algorithm called *computeBCov* for computing the best covers of a concept Q using a terminology $\mathcal{T}$. In the previous section, we showed that this problem can be reduced to the search of the transversals with minimal cost of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. The problem of computing minimal transversals of a hypergraph is central in various fields of computer science [16]. The precise complexity of this problem is still an open issue. In [19], it is shown that the generation of the transversal hypergraph can be done in incremental subexponential time $k^{O(logk)}$, where k is the combined size of the input and the output. To the best of our knowledge, this is the best time bound for the problem of computing minimal transversals of a hypergraph.

A classical algorithm for computing the minimal transversals of a hypergraph is presented in [7,16,27]. The algorithm is incremental and works in n steps, where n is the number of edges of the hypergraph. Starting from an empty set of transversals, the basic idea is to explore each edge of the hypergraph, one edge in each step, and to generate a set of candidate transversals by computing all the possible unions of the candidates generated in the previous step and each vertex in the considered edge. At each step, the nonminimal candidate transversals are pruned.

Thus a naive approach to computing the minimal transversals with a minimal cost consists in computing all the minimal transversals, using such an algorithm, and then choosing those transversals that have the minimal cost. The *computeBCov* algorithm presented below makes the following improvements with respect to the naive approach:

1. It reduces the number of candidates in the intermediary steps of the algorithm by generating only the minimal transversals.
2. It uses a combinatorial optimization technique (branch-and-bound) in order to prune, at the intermediary steps of the algorithm, those candidate transversals that will not generate transversals with a minimal cost.

These two optimizations have been implemented as separate options of the algorithm, namely, option Pers for the first optimization and option BnB for the second one.[8] The

[8] Other less significant optimization options have also been implemented.

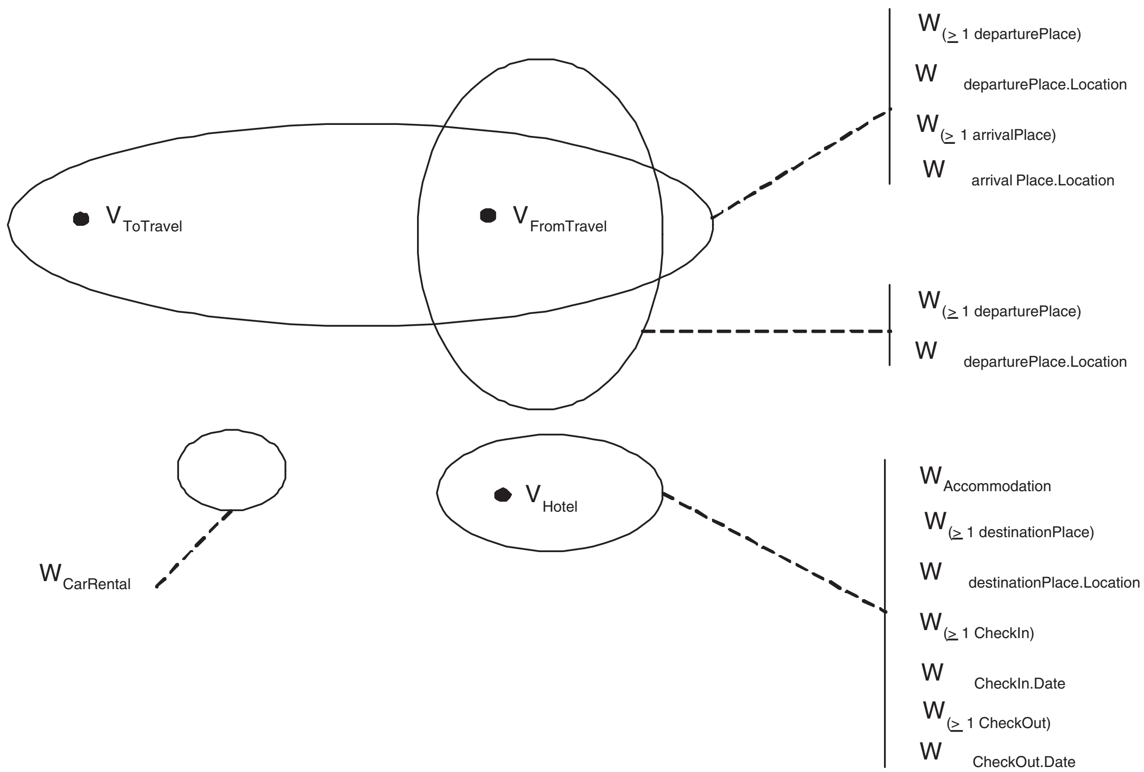

Fig. 1. Example of a hypergraph

first optimization allows one to generate only good candidates (only minimal transversals) at each iteration (line 5 of the algorithm). To do so, we use a necessary and sufficient condition (provided by Theorem 2 below) to describe a pair (X_i, s_j) that will generate a nonminimal transversal at iteration i, where X_i is a minimal transversal generated at iteration $i-1$ and s_j is a vertex of the i-th edge.

Algorithm 1 ***computeBCov*** (skeleton)

Require: An instance $\mathcal{BCOV}(\mathcal{T}, Q)$ of the best covering problem.
Ensure: The set of the best covers of Q using $\mathcal{T}$.
1: Build the associated hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$.
2: $Tr \leftarrow \emptyset$ – Initialization of the minimal transversal set.
3:
$CostEval \leftarrow \sum_{e \in \Gamma'} \min_{V_{S_i} \in e} (|Miss_{S_i}(Q)|)$. – Initialization of CostEval
4: **for all** edge $E \in \Gamma'$ **do**
5: $Tr \leftarrow$ the newly generated set of the minimal transversals.
6: Remove from Tr the transversals whose costs are greater than $CostEval$.
7: Compute a more precise evaluation of $CostEval$.
8: **end for**
9: **for all** $X \in Tr$ such that $|Miss_{E_X}(Q)| = CostEval$ **do**
10: return the concept E_X as a best cover of Q using $\mathcal{T}$.
11: **end for**

Theorem 2 *Let $Tr(\mathcal{H}) = \{X_i, i = 1..m\}$ be the set of minimal transversals for the hypergraph $\mathcal{H}$ and $E = \{s_j, j = 1..n\}$ an edge of $\mathcal{H}$. Assume $\mathcal{H}' = \mathcal{H} \cup E$. Then we have: $X_i \cup \{s_j\}$ is a nonminimal transversal of $\mathcal{H}'$ $\Leftrightarrow$ there exists a minimal transversal X_k of $\mathcal{H}$ such that $X_k \cap E = \{s_j\}$ and $X_k \setminus \{s_j\} \subset X_i$.*

Details and proofs of Theorem 2 are given in [33].

The second improvement consists in a branch-and-bound like enumeration of transversals. First, a simple heuristic is used to efficiently compute the cost of a *good* transversal (i.e., a transversal expected to have a small cost). This can be carried out by adding, for each edge of the hypergraph, the cost of the vertex that has the minimal cost. The resulting cost is stored in the variable $CostEval$ (line 3 of the algorithm). Recall that for any set of vertices $X = \{S_i, \ldots, S_n\}$:

$$cost(X) = |Pmiss_{S_i \sqcap \ldots \sqcap S_n}(Q)| \leq \sum_{j \in [i,n]} |Pmiss_{S_j}(Q)|$$
$$= \sum_{S_j \in X} cost(S_j).$$

The evaluation of $CostEval$ is an upper bound of the cost of an existing transversal. As we consider candidates in intermediate steps of the algorithm, we can eliminate from $Tr(\mathcal{H}_{\mathcal{T}Q})$ any candidate transversal that has a cost greater than $CostEval$ since that candidate could not possibly lead to a transversal that is better than what we already have (line 6). From each candidate transversal remaining in $Tr(\mathcal{H}_{\mathcal{T}Q})$ we compute a new evaluation for $CostEval$ by considering only remaining edges (line 7).

At the end of the algorithm, each computed minimal transversal $X \in Tr$ is transformed into a concept E_X that constitutes an element of the solution to the $\mathcal{BCOV}(\mathcal{T}, Q)$ problem (line 10).

5 Semantic reasoning for Web services discovery

In this section, we describe how the proposed reasoning mechanism can be used to automate the discovery of Web services in the context of DAML-S ontologies.[9] More details on these aspects can be found in [5].

DAML-S [13] is an ontology for describing Web services. It employs the DAML+OIL ontology language [14] to describe the properties and capabilities of Web services in a computer-interpretable form, thereby facilitating the automation of Web service discovery, invocation, composition and execution. DAML-S supplies a core set of markup language constructs for describing Web services in terms of classes (concepts) and complex relationships between them.

A DAML-S ontology of services is structured in three main parts [13]:

- *ServiceProfile* describes the capabilities and parameters of the service. It is used for advertising and discovering services.
- *ServiceModel* gives a detailed description of a service's operation. Service operation is described in terms of a process model that presents both the control structure and data flow structure of the service required to execute a service.
- *ServiceGrounding* specifies the details of how to access the service via messages (e.g., communication protocol, message formats, addressing, etc).

The service profile provides information about a service that can be used by an agent to determine if the service meets its needs. It consists of three types of information: a (human-readable) *description* of the service, the *functional behavior* of the service that is represented as a transformation from the service inputs to the service outputs, and several *nonfunctional attributes* that specify additional information about a service (e.g., the cost of the service).

In the DAML-S approach, a service profile is intended to be used by providers to advertise their services as well as by service requesters to specify their needs.

5.1 Best covering profile descriptions

We describe now how the proposed algorithm can be adapted to support dynamic discovery of DAML-S services. It is worth noting that we do not deal with the full expressiveness of the DAML+OIL language. We consider only DAML-S ontologies expressed using a subset of the DAML+OIL language for which a structural subsumption algorithm exists. Below such ontologies are called *restricted DAML-S ontologies*.

As proposed in [31], a match between a query (expressed by means of a service profile) and an advertised service is determined by comparing all the outputs of the query with the outputs of the advertisement and all the inputs of the advertisement with the inputs of the query. We adopt the same idea for comparing requests with advertised services, but we propose to use *computeBCov* instead of the matchmaking algorithm given in [31]. Intuitively, we target a service discovery mechanism that works as follows: given a service request Q and a DAML-S ontology $\mathcal{T}$, we want to compute the best combination of Web services that satisfies *as much as possible* the outputs of the request Q and that requires *as few input as possible* that are not provided in the description of Q. We call such a combination of Web services a *best profile cover* of Q using $\mathcal{T}$. To achieve this task, we need to extend the best covering techniques, as presented in Sect. 3, to take into account profile descriptions as presented below.

Let $\mathcal{T} = \{S_i, i \in [1, n]\}$ be a restricted DAML-S ontology and $E \equiv S_l \sqcap \ldots \sqcap S_p$, with $l, p \in [1, n]$, be a conjunction of some services occurring in $\mathcal{T}$. We denote by $I(E)$ (resp. $O(E)$) the concept determined using the conjunction of all the inputs (resp. the outputs) occurring in the profile section of all the services S_i, for all $i \in [l, p]$. In the same way, we use $I(Q)$ (resp. $O(Q)$) to denote the concept determined using the conjunction of all the inputs (resp. the outputs) occurring in the profile section of a given query Q.

We extend the notions of cover, rest, and miss to service profiles as follows.

Definition 11 ***Profile cover (Pcover)***
A profile cover, called Pcover, of Q using $\mathcal{T}$ is a conjunction E of some services S_i from $\mathcal{T}$ such that $O(Q) - lcs_{\mathcal{T}}(O(Q), O(E)) \not\equiv O(Q)$.

Hence, a Pcover of a query Q using $\mathcal{T}$ is defined as any conjunction of Web services occurring in $\mathcal{T}$ that shares some outputs with Q.

Definition 12 ***Profile rest (Prest) and profile miss (Pmiss)***
Let Q be a service request and E a cover of Q using $\mathcal{T}$. The Prest of Q with respect to E, denoted by $Prest_E(Q)$, is defined as follows: $Prest_E(Q) \doteq O(Q) - lcs_{\mathcal{T}}(O(Q), O(E))$. The profile missing information about Q with respect to E, denoted by $Pmiss_E(Q)$, is defined as follows: $Pmiss_E(Q) \doteq I(E) - lcs_{\mathcal{T}}(I(Q), I(E))$.

Finally, the notion of best profile cover can be extended to profiles by respectively replacing *rest* and *miss* by *Prest* and *Pmiss* in definition 7 [6]. Consequently, the algorithm *computeBCov*, presented in the previous section, can be adapted and used as a matchmaking algorithm for discovering DAML-S services based on their capabilities. We devised a new algorithm, called *computeBProfileCov*, for this purpose. According to definitions 11 and 12, the algorithm selects the combinations of services that best match a given query and effectively computes the outputs of the query that cannot be satisfied by the available services (i.e., *Prest*) as well as the inputs that are required by the selected services and are not provided in the query (i.e., *Pmiss*).

5.2 Illustrative example

This example illustrates how the notion of best profile cover can be used to match a service request with service advertisements. Let us consider an ontology of Web services that contains the following three services:

[9] http://www.daml.org/services/

Table 3. Input and output service parameters

Service	Input	Output
ToTravel	Itinerary, Arrival	TripReservation
FromTravel	Itinerary, Departure	TripReservation
Hotel	Destination, StayDuration	HotelReservation

- $ToTravel$ allows one to reserve a trip given an itinerary (i.e., the departure point and the arrival point) and the arrival time and date.
- $FromTravel$ allows one to reserve a trip given an itinerary and the departure time and date.
- $Hotel$ allows one to reserve a hotel given a destination place, a period of time expressed in terms of the check-in date, and the check-out date.

Table 3 shows the input and the output concepts of the three Web services. We assume that the service profiles refer to concepts defined in the restricted DAML+OIL tourism ontology given in Table 4. For the sake of clarity, we use the usual DL syntax instead of the DAML+OIL syntax to describe the ontology. In Table 4, the description of the concept Itinerary denotes the class of individuals whose departure places (resp. arrival places) are instances of the concept Location. Moreover, the individual that belongs to this class must have at least one departure place (the constraint (≥ 1 departurePlace)) and at least one arrival place (the constraint (≥ 1 arrivalPlace)). The input of the service ToTravel is obtained using the conjunction of all its inputs as follows: $\mathtt{I(ToTravel)} \equiv \mathtt{Itinerary} \sqcap \mathtt{Arrival}$. By replacing the concepts Itinerary and Arrival with their descriptions, we obtain the following equivalent description:

$I(ToTravel) \equiv$ ($\geq$ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location) $\sqcap$ ($\geq$ 1 arrivalDate) $\sqcap$ ($\forall$ arrivalDate.Date) $\sqcap$ ($\geq$ 1 arrivalTime) $\sqcap$ ($\forall$ arrivalTime.Time)

The inputs and outputs of the other Web services can be computed in the same way.

Now let us consider a service request Q that searches 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 check-in and check-out dates. The inputs and outputs of the query Q can be expressed by the following descriptions that, again, refer to some concepts of the tourism ontology given in Table 4:

$I(Q) \equiv$ ($\geq$ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location) $\sqcap$ ($\geq$ 1 departureDate) $\sqcap$ ($\forall$ departureDate.Date) $\sqcap$ ($\geq$ 1 destinationPlace) $\sqcap$ ($\forall$ destinationPlace.Location) $\sqcap$ ($\geq$ 1 checkIn) $\sqcap$ ($\forall$ checkIn.Date) $\sqcap$ ($\geq$ 1 checkOut) $\sqcap$ ($\forall$ checkOut.Date)

$O(Q) \equiv$ TripReservation $\sqcap$ HotelReservation $\sqcap$ CarRental

The matching between the service request Q and the three advertised services given above can be achieved by computing the best profile cover of Q using these services. The result is the following:

Best profile cover	Prest	Pmiss
FromTravel, Hotel	CarRental	departureTime

In this example, there is only one best profile cover of Q corresponding to the description $E \equiv Hotel \sqcap FromTravel$. The selected services generate the concepts $TripReservation$ and $HotelReservation$, which are part of the output required by the query Q. From the service descriptions we can see that no Web service supplies the concept CarRental. Hence, the best profile covers of Q will have exactly the following profile rest: $Prest_E(Q) \equiv carRental$. This rest corresponds to the output of the query that cannot be generated by any advertised service. Moreover, the **Pmiss** column shows the information (the role departureTime) required as input of the selected services but not provided in the query inputs. More precisely, the best profile covers of Q will have exactly the following profile missing information: $Pmiss_E(Q) \equiv$ *($\geq$ 1 departureTime) $\sqcap$ ($\forall$ departureTime.Time)*. It is worth noting that, although the solution $E' \equiv Hotel \sqcap ToTravel$ generates the same outputs (i.e., the concepts $TripReservation$ and $HotelReservation$), it will not be selected because its **Pmiss** is greater than that of the first solution (it contains the roles arrivalTime and arrivalDate).

6 Evaluation and experiments

In this section, we describe a testbed prototype implementation of the *computeBCov* algorithm. This prototype implementation has been motivated by three goals: (i) to validate the feasibility of the approach, (ii) to test the correctness of the *computeBCov* algorithm, and (iii) to study the performance and scalability of *computeBCov*.

The first two goals were achieved in the context of a European project – the MKBEEM project.[10] To achieve the third goal, we have integrated into the prototype a tool based on the IBM XML Generator (http://www.alphaworks.ibm.com/tech/xmlgenerator) that enables one to generate random XML-based service ontologies and associated service requests.

6.1 Application scenario

We used our prototype in the context of the MKBEEM project, which aims at providing electronic marketplaces with intelligent, knowledge-based multilingual services. The main objective of MKBEEM is to create an *intelligent knowledge-based multilingual* mediation service that features the following building blocks [29]:

- Natural language interfaces for both the end user and the system's content providers/service providers.
- Automatic multilingual cataloging of products by service providers.
- Online e-commerce contractual negotiation mechanisms in the language of the user that guarantee safety and freedom.

In this project, ontologies are used to provide a consensual representation of the electronic commerce field in two typical domains (tourism and mail order). In MKBEEM, ontologies are structured in three layers, as shown in Fig. 2. The global

[10] MKBEEM stands for Multilingual Knowledge Based European Electronic Marketplace (IST-1999-10589, 1 February 2000–1 December 2002): *http://www.mkbeem.com*.

Table 4. Example of a tourism ontology

Itinerary	$\equiv$	(≥ 1 departurePlace) $\sqcap$ ($\forall$ departurePlace.Location) $\sqcap$ ($\geq$ 1 arrivalPlace) $\sqcap$ ($\forall$ arrivalPlace.Location)
Arrival	$\equiv$	($\geq$ 1 arrivalDate) $\sqcap$ ($\forall$ arrivalDate.Date) $\sqcap$ ($\geq$ 1 arrivalTime) $\sqcap$ ($\forall$ arrivalTime.Time)
Departure	$\equiv$	($\geq$ 1 departureDate) $\sqcap$ ($\forall$ departureDate.Date) $\sqcap$ ($\geq$ 1 departureTime) $\sqcap$ ($\forall$ departureTime.Time)
Destination	$\equiv$	($\geq$ 1 destinationPlace) $\sqcap$ ($\forall$ destinationPlace.Location)
StayDuration	$\equiv$	($\geq$ 1 checkIn) $\sqcap$ ($\forall$ checkIn.Date) $\sqcap$ ($\geq$ 1 checkOut) $\sqcap$ ($\forall$ checkOut.Date)
TripReservation	$\equiv$	...
HotelReservation	$\equiv$	...
CarRental	$\equiv$	...

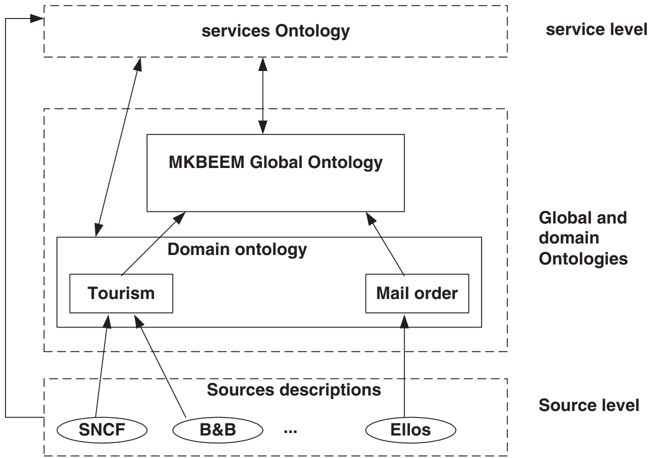

Fig. 2. Knowledge representation in the MKBEEM system

ontology describes the common terms used in the whole MKBEEM platform. This ontology represents the general knowledge in different domains while each domain ontology contains specific concepts corresponding to vertical domains (e.g., tourism). The service ontology describes classes of services, e.g., service capabilities, nonfunctionnal attributes, etc. The source descriptions specify concrete services (i.e., provider offerings) in terms of service ontology. The MKBEEM mediation system allows one to fill the gap between customer queries (possibly expressed in a natural language) and diverse concrete provider offerings. In a typical scenario, users express their requests in a natural language, and the requests are then translated into ontological formulas expressed using domain ontologies. Then, the MKBEEM system relies on the proposed reasoning mechanism to reformulate user queries against the domain ontology in terms of Web services. The aim here is to allow the users/applications to automatically discover the available Web services that best meet their needs, to examine service capabilities, and to possibly complete missing information.

In our implementation we used ontologies with approximately 300 concepts and 50 Web services to validate the applicability of the proposed approach. Indeed, this implementation has shown the effectiveness of the proposed matchmaking mechanism in two distinct end-user scenarios: (i) business-to-consumer online sales and (ii) Web-based travel/tourism services.

6.2 Quantitative experiments

To conduct experiments, we have implemented up to six versions of the *computeBCov* algorithm corresponding to different combinations of optimization options. The prototype is implemented using the Java programming language. All experiments were performed using a PC with a 500-MHz Pentium III and 384 MB RAM.

To quantitatively test *computeBCov*, we first have to run *computeBCov* on the worst cases and then on a set of ontologies and queries randomly generated by our prototype. *computeBCov* worst cases were built according to a theoretical study of the complexity of all versions of *computeBCov*: two ontologies (and their associated queries) were built to maximize the number of minimal transversals of the corresponding hypergraph as well as the number of elementary operations of the algorithm (i.e., inclusion tests and intersection operations). In each case, there exists at least one version of *computeBCov* that completes the execution in less than 20 s. It should be noted that although these cases are bad for *computeBCov*, they are also totally unrealistic with respect to practical situations.

We generated larger but still realistic random ontologies. We focus here on three case studies with varying sizes of application domain ontology, of the Web service ontology, and of the query. Their characteristics are given below:

Configurations	**Case 1**	**Case 2**	**Case 3**
Number of defined concepts in the application domain ontology	365	1334	3405
Number of Web services	366	660	570
Number of (atomic) clauses in the query	6	33	12

Note that the internal structures of these ontologies correspond to bad cases for the *computeBCov* algorithm. We have run the six versions of the *computeBCov* algorithm based on these

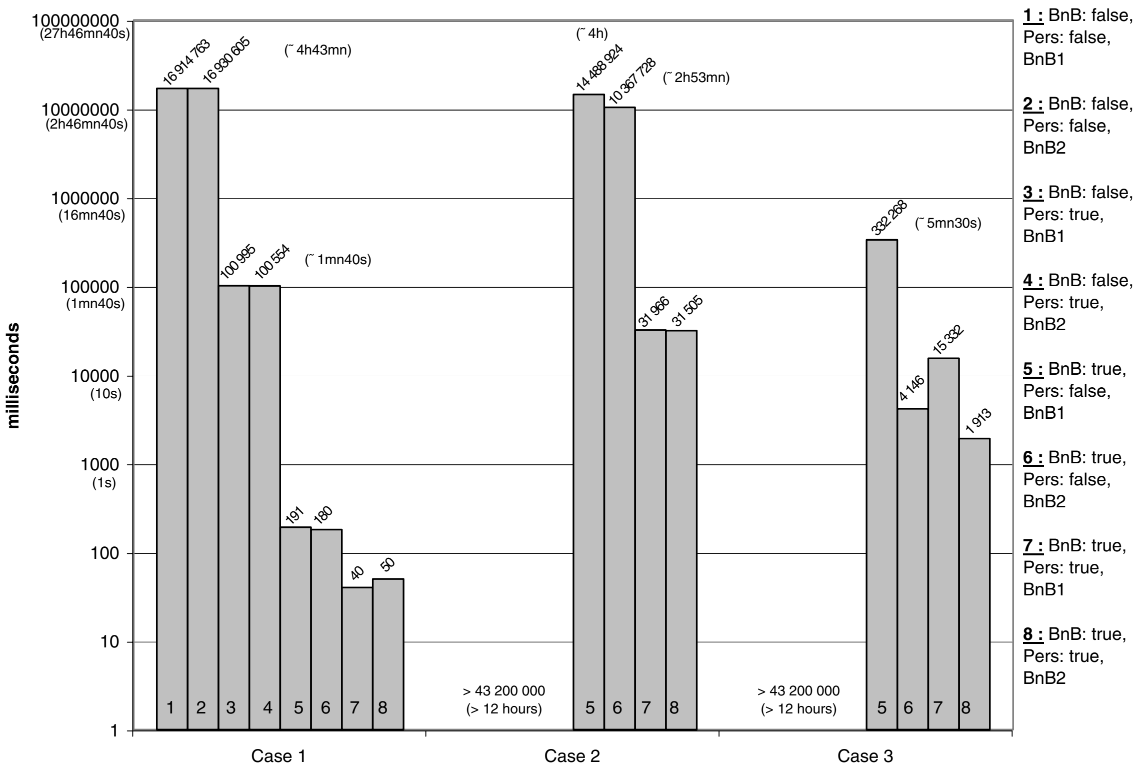

Fig. 3. Execution time

cases. The overall execution time results are given in Fig. 3.[11] This figure shows that for cases 1 and 3 (resp. case 2), there is at least a version of the algorithm that runs in less than 2 s (resp. less than 30 s). Although Fig. 3 shows that there is a great difference in performance of the different versions of the algorithm, in each case there is at least one efficient version of the algorithm even when the domain ontology is quite large. Details about the implementation of *computeBCov*, the theoretical study of worst cases, the parameterized ontology generation process, and experimental results can be found in [33].

7 Related work

In this section, we first review related work in the area of Semantic Web services discovery, then we examine the relationship of our work with the problem of query (concept) rewriting.

7.1 Semantic Web services discovery

Current Web services infrastructure have serious limitations with respect to meeting the automation challenges. For example, UDDI provides limited search facilities allowing only a keyword-based search of businesses, services, and the so-called tModels based on names and identifiers. To cope with this limitation, emerging approaches rely on Semantic Web technology to support service discovery [21,31]. For example, [8] proposes to use process ontologies to describe the behavior of services and then to query such ontologies using a process query language (PQL). Chakraborty et al. [11] define an ontology based on DAML [12] to describe mobile devices and propose a matching mechanism that locates devices based on their features (e.g., a type of a printer). The matching mechanism exploits rules that use the ontology, service profile information, and query to perform matching based on relationships between attributes and their values. A Prolog-based reasoning engine is used to support such a matching. There are other approaches based on a DAML+OIL [25] description of services that propose to exploit the DL-based reasoning mechanisms.

An experience in building a matchmaking prototype based on a DL reasoner that considers DAML+OIL-based service descriptions is reported by [21]. The proposed matchmaking algorithm is based on simple subsumption and consistency tests. A more sophisticated matchmaking algorithm between services and requests described in DAML-S is proposed by [31].[12] The algorithm considers various degrees of matching that are determined by the minimal distance between concepts in the concept taxonomy. Based on a similar approach, the ATLAS matchmaker [32] considers DAML-S ontologies and utilizes two separate sets of filters: (1) Matching functional attributes to determine the applicability of advertisements (i.e., do they deliver sufficient quality of service, etc). The matching is achieved by performing conjunctive pairwise comparison for the functional attributes; (2) matching service functionality to determine if the advertised service matches

[11] Note that versions 1 and 2 of the algorithm (resp. 3 and 4) are similar as both run *computeBCov* without BnB, and what distinguishes 1 from 2 (resp. 3 from 4) is the way the option BnB is implemented (BnB1 or BnB2).

[12] http://www.daml.org/services/

the requested service. A DAML-based subsumption inference engine is used to compare input and output sets of requests and advertisements.

Finally, it should be noted that the problem of capabilities-based matching has also been addressed by several other research communities, e.g., information retrieval, software reuse systems, and multiagent communities. More details about these approaches and their applicability in the context of the Semantic Web services area can be found in [8,31]. Our work makes complementary contributions to the efforts mentioned above. More precisely, our approach builds upon existing service description languages and provides the following building blocks for flexible and effective service discovery:

- A global reasoning mechanism: we propose a matchmaking algorithm that goes beyond a pairwise comparison between a service request and service offerings by allowing the discovery of *combinations* of services that match (cover) a given request.
- A flexible matchmaking process that goes beyond subsumption tests.
- Effective computation of missed information: the difference between the query and its rewriting (i.e., rest and miss) is effectively computed and can be used, for example, to improve service repository interactions.

7.2 Query (concept) rewriting

From a technical point of view, the best covering problem belongs to the general framework for *rewriting using terminologies* provided in [3]. This framework is defined as follows:

- Given a terminology $\mathcal{T}$ (i.e., a set of concept descriptions), a concept description Q that does not contain concept names defined in $\mathcal{T}$ and a binary relation ρ between concept descriptions, can Q be rewritten into a description E, built using (some) of the names defined in $\mathcal{T}$, such that $Q\rho E$?
- Additionally, some optimality criterion is defined to select the relevant rewritings.

Already investigated instances of this problem are the minimal rewriting problem [3] and rewriting queries using views [4,20,23].

Minimal rewriting is concerned with the problem of rewriting a concept description Q into a shorter but *equivalent* description (hence ρ is equivalence modulo $\mathcal{T}$ and the size of the rewriting is used as the optimality criterion). Here the focus is on determining a rewriting that is shorter and more readable than the original description.

The problem of rewriting queries using views has been intensively investigated in the database area (see [23] for a survey). The purpose here is to rewrite a query Q into a query expression that uses only a set of views $\mathcal{V}$. Two main kinds of rewritings have been studied:

- Maximally contained rewritings where ρ is the subsumption and the optimality criterion is the inverse subsumption. This kind of rewriting plays an important role in many applications such as information integration and data warehousing.
- Equivalent rewriting where ρ is the equivalence and the optimality criterion is minimization of the cost of the corresponding query plan. This kind of rewriting has been used mainly for query optimization purposes.

The *best covering problem* can be viewed as a new instance of the problem of rewriting concepts using terminologies where:

- ρ correponds to the notion of cover (hence, it is neither equivalence nor subsumption), and
- The optimality criterion is the minimization of the rest and the miss.

8 Conclusion

In this paper we have presented a novel approach to automate the discovery of Web services. We formalized service discovery as a rewriting process and then investigated this problem in the context of restricted framework of description logics with structural subsumption. These logics ensure that the difference operation is always semantically unique and can be computed using a structural difference operation. In this context, we have shown that the best covering problem can be mapped to the problem of computing the minimal transversals with minimum cost of a “weighted" hypergraph.

The framework of languages with a semantically unique difference appears to be sufficient in the context of the MKBEEM project. But the languages that are proposed to achieve the Semantic Web vision appear to be more expressive. Our future work will be devoted to the extension of the proposed framework to hold the definition of the best covering problem for description logics where the difference operation is not semantically unique. In this case, the difference operation does not yield a unique result and thus the proposed definition of a best cover is no longer valid. We also plan to (i) consider service discovery when there is a large number of heterogeneous ontologies and (ii) extend the proposed technique to support service composition automation.

References

1. Baader F, Calvanese D, McGuinness D, Nardi D (2003) Patel-Schneider P (ed) The description logic handbook: theory, implementation and applications. Cambridge University Press, Cambridge, UK
2. Baader F, Küsters R, Molitor R (1999) Computing least common subsumer in description logics with existential restrictions. In: Dean T (ed) Proceedings of the 16th international joint conference on AI, Stockholm, Sweden, 31 July–6 August 1999, pp 96–103
3. Baader F, Küsters R, Molitor R (2000) Rewriting concepts using terminologies. In: Proceedings of the 7th international conference on principles of knowledge representation and reasoning (KR’2000), Colorado, April 2000, pp 297–308
4. Beeri C, Levy AY, Rousset M-C (1997) Rewriting queries using views in description logics. In: Yuan L (ed) Proceedings of ACM PODS, April 1997, New York, pp 99–108
5. Benatallah B, Hacid M-S, Rey C, Toumani F (2003a) Request rewriting-based Web service discovery. In: Fensel D, Sycara

K, Mylopoulos J (eds) Proceedings of the international Semantic Web conference (ISWC 2003), Sanibel Island, FL, October 2003. Lecture notes in computer science, vol 2870. Springer, Berlin Heidelberg New York, pp 242–257
6. Benatallah B, Hacid M-S, Rey C, Toumani F (2003b) Semantic reasoning for Web services discovery. In: Proceedings of the WWW workshop on e-services and the Semantic Web, Budapest, Hungary, May 2003
7. Berge C (1989) Hypergraphs. In: North Holland Mathematical Library, vol 45. Elsevier, North-Holland
8. Bernstein A, Klein M (2002) Discovering services: towards high precision service retrieval. In: Proceedings of the CaiSE workshop on Web Services, e-business, and the Semantic Web: foundations, models, architecture, engineering and applications, Toronto, May 2002
9. Casati F, Shan M-C, Georgakopoulos D (eds) (2001) J Very Large Databases Special Issue E-Serv 10(1):117
10. Casati F, Shan M-C (2001) Dynamic and adaptive composition of e-services. Inf Sys 26(3):143–163
11. Chakraborty D, Perich F, Avancha S, Joshi A (2001) DReggie: semantic service discovery for M-Commerce applications. In: Proceedings of the workshop on reliable and secure applications in mobile environment, 20th symposium on reliable distributed systems, New Orleans, October 2001, pp 28–31
12. DAML Services. http://www.daml.org/services/
13. DAML Services Coalition (2002) DAML-S: Web service description for the Semantic Web. In: Proceedings of the 1st international Semantic Web conference (ISWC), Sardinia, Italy, June 2002, pp 348–363
14. Ding Y, Fensel D, Omelayenko B, Klein MCA (2002) The Semantic Web: yet another hip? Data Knowl Eng 6(2–3):205–227
15. Donini F, Schaerf A, Lenzerini M, Nardi D (1996) Reasoning in description logics. In: Brewka G (ed) Foundation of knowledge representation. CSLI-Publications, Stanford, CA, pp 191–236
16. Eiter T, Gottlob G (1995) Identifying the minimal transversals of a hypergraph and related problems. SIAM J Comput 24(6):1278–1304
17. Fensel D, Bussler C, Ding Y, Omelayenko B (2002) The Web Service Modeling Framework WSMF. Electron Commerce Res Appl 1(2): 113–137
18. Fensel D, Bussler C, Maedche A (2002) Semantic Web enabled Web services. In: Proceedings of the international Semantic Web conference, Sardinia, Italy, June 2002, pp 1–2
19. Freidman ML, Khachiyan L (1996) On the complexity of dualization of monotone disjunctive normal forms. J Algorithms 21:618–628
20. Goasdoué F, Rousset M-C, Lattès V (2000) The use of CARIN language and algorithms for information integration: the PICSEL system. Int J Cooper Inf Sys 9(4):383–401
21. González-Castillo J, Trastour D, Bartolini C (2001) Description logics for matchmaking of services. In: Proceedings of the KI-2001 workshop on applications of description logics, Vienna, Austria, September 2001. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-44/
22. Hacid M-S, Leger A, Rey C, Toumani F (2002) Dynamic discovery of e-services: a description logics based approach. Report, LIMOS, Clemont-Ferrand, France. http://www.710.univ-lyon1.fr/∼dbkrr/publications.htm
23. Halevy AY (2001) Answering queries using views: a survey. J Very Large Databases 10(4):270–294
24. Hendler J, McGuinness DL (2000) The DARPA Agent Markup Language. IEEE Intell Sys 15(6):67–73
25. Horrocks I (2002) DAML+OIL: a reasonable Web ontology language. In: Proceedings of EDBT'2002, Prague, Czech Republic, March 2002, pp 2–13
26. Horrocks I, Patel-Schneider PF, van Harmelen F (2002) Reviewing the design of DAML+OIL: an ontology language for the Semantic Web. In: Proceedings of the 18th national conference on artificial intelligence (AAAI 2002), Edmonton, Alberta, Canada, 28 July–1 August 2002, pp 792–797
27. Mannila A, Räihä K-J (1994) The design of relational databases. Addison-Wesley, Wokingham, UK
28. McIlraith S, Son TC, Zeng H (2001) Semantic Web services. IEEE Intell Sys Special Issue Semantic Web 16(2):46–53
29. MKBEEM (2002) http://www.mkbeem.com
30. Molitor R (1998) Structural subsumption for $\mathcal{ALN}$. LTCS-Report LTCS-98-03, LuFG Theoretical Computer Science, RWTH Aachen, Germany
31. Paolucci M, Kawamura T, Payne TR, Sycara KP (2002) Semantic matching of Web services capabilities. In: Proceedings of the international Semantic Web conference, Sardinia, Italy, June 2002, pp 333–347
32. Payne TR, Paolucci M, Sycara K (2001) Advertising and matching DAML-S service descriptions (position paper). In: Proceedings of the international Semantic Web working symposium, Stanford, CA, July 2001
33. Rey C, Toumani F, Hacid M-S, Leger A (2003) An algorithm and a prototype for the dynamic discovery of e-services. Technical Report RR05-03, LIMOS, Clemont-Ferrand, France. http://www.isima.fr/limos/publications.htm
34. Teege G (1994) Making the difference: a subtraction operation for description logics. In: Doyle J, Sandewall E, Torasso P (eds) Proceedings of KR'94, location, day month 1994. Morgan Kaufmann, San Francisco
35. Weikum G (ed) (2001) Data Eng Bull Special Issue Infrastruct Adv E-Serv 24(1). IEEE Press, New York
36. World Wide Web Consortium (2003) http://www.w3.org/2001/sw/webont/