Dynamic discovery of e-services

A description logics based approach

Mohand-Said Hacid[*] — Alain Léger[**] —
Christophe Rey[***] — Farouk Toumani[***]

[*] *Laboratoire d'Ingénierie des Systèmes d'Information*
UFR d'Informatique, Université Claude Bernard Lyon I
Bâtiment Nautibus 8, boulevard Niels Bohr, F-69622 Villeurbanne cedex
mshacid@lisi.fr

[**] *France Telecom R&D/DMI*
BP 59, 4 rue du clos courtel F-35512 Cesson Sévigné
alain.leger@rd.francetelecom.com

[***] *Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes*
CNRS UMR 2239 – Université Blaise-Pascal Clermont-Ferrand II
24 avenue des Landais F-63177 Aubière cedex
{rey,ftoumani}@isima.fr

ABSTRACT. *We investigate the problem of dynamic discovery of e-services, important in the loosely coupled vision of e-commerce. We show that it can be viewed as a new important reasoning technique in Description Logics: it can be viewed as a new instance of the problem of rewriting concepts using terminologies which generalizes problems such as rewriting queries using views or minimal rewriting using terminologies. We call this new instance the best covering using terminology: given a query Q and a set $\mathcal{S}_\mathcal{T}$ of e-services, the problem consists in finding a subset of $\mathcal{S}_\mathcal{T}$ called a "cover" of Q that contains as much as possible of common information with Q and as less as possible of extra information with respect to Q. We formally study this problem for languages with structural subsumption. We show that it is **NP-hard** and propose an algorithm derived from hypergraphs theory.*

RÉSUMÉ. *Nous étudions le problème de la découverte dynamique de e-services, important dans l'approche faiblement couplée du e-commerce. Nous montrons qu'il peut être vu comme un raisonnement nouveau dans le domaine des logiques de description : il s'agit alors d'une nouvelle instance du problème de la réécriture de concepts en utilisant une terminologie, d'autres instances étant la réécriture de requêtes en utilisant des vues ou encore la recherche de réécritures minimales en utilisant une terminologie. Nous appelons cette nouvelle instance la recherche des meilleures couvertures en utilisant une terminologie : soit une requête Q et un ensemble $\mathcal{S}_\mathcal{T}$ de e-services, il s'agit d'identifier les sous-ensembles de $\mathcal{S}_\mathcal{T}$, que l'on appellera "couvertures" de Q, qui contiennent le plus possible d'informations communes avec Q et le moins* *possible d'informations absentes de Q. Nous étudions le problème au niveau formel pour des langages à subsomption structurelle. Nous montrons qu'il est **NP-hard**. Enfin nous proposons un algorithme issu de la théorie des hypergraphes pour le résoudre.*

KEYWORDS: *E-services discovery, Description Logics, covers of concepts, rewriting, difference, hypergraphs transversals*

MOTS-CLÉS : *Découverte de e-services, Logiques de Description, couvertures d'un concept, réécriture, différence, transversaux d'un hypergraphe*

1. Introduction

The recent progress and wider dissemination of electronic commerce via the World Wide Web is revolutionizing the way companies interact with their suppliers, partners or clients. The number and type of on-line resources and services increased considerably and lead to a new form of automation, namely B2B and B2C e-commerce. A recent initiative envisions a new paradigm for electronic commerce in which applications are wrapped and presented as integrated electronic services (e-services) [Wei01, Vld01]. Roughly speaking, an e-service (also called Web service) can be defined as an application made available via the Internet by a service provider, and accessible by clients [CAS 01b, BEN 02]. Examples of e-services currently available range from weather forecast, on-line travel reservation or banking services to entire business functions of an organization. The ultimate vision behind the e-service paradigm is to transform the Web from a collection of information into a distributed device of computation where programs (services) are capable of intelligent interaction by being able to discover and negotiate with each other and compose themselves into more complex services [CAS 01a, Wei01, FEN 02, BEN 02]. Automation is a key concept to realize this vision. It is fundamental at each step of the service delivery to cope with the highly dynamic environment of e-services [CAS 01a, Vld01].

This paper focuses on the problem of dynamic discovery of e-services. Such a process involves automatic matching of service offers with service requests and constitutes an important aspect of e-commerce interactions. For example, it is the first step to enable automated service composition. Our aim is to ground the dynamic discovery of e-services on a semantic comparison between a client query and available e-services to provide the *combinations* of e-services that *“best match"* the client needs. However, to achieve such an advanced discovery process two main issues must be addressed [PAO 02]:

– Description of services: an automated discovery process requires rich and flexible *machine understandable* descriptions of services that are not supported by the current industry standards (e.g., UDDI[1]).

– An algorithm that allows to reason about the description of e-services to achieve the discovery task.

It is worth noting that the semantic web initiative[2] at W3C aims at generating technologies and tools that might help bridge the gap between the current standards solutions and the requirement of and advanced e-services discovery process [PAO 02, FEN 02]. In this context, ontologies can play a crucial role to define formal semantics for information [FEN 01, FEN 02, HOR 02b], consequently allowing computer-interpretable specifications of services. In the line of the semantic web approach, our work rests on a knowledge representation approach to allow a rich description of e-services and to provide adequate reasoning mechanisms that automate the discovery

1. Universal Description, Discovery and Integration (http://www.uddi.org/).
2. http://www.w3.org/2001/sw/.

of e-services. We propose to use description logics (DLs) [DON 96] as a description language to specify the declarative part of e-services. A key aspect of description logics is their formal semantics and reasoning support. They have proven to provide a useful support for the definition, integration and maintenance of ontologies- a feature that makes them suitable for the semantic Web [HEN 00, HOR 02b, HOR 02a].

Problem statement and contributions This paper concentrates on the reasoning issue to automate the discovery of e-services. In this setting, the problem of dynamic discovery of e-services can be stated as follows: given an ontology $\mathcal{T}$ containing e-services descriptions and a client query Q, find a combination of e-services that contains as much as possible of *common* information with Q and as less as possible of *extra information* with respect to Q. We call such a combination of e-services a *best cover* of Q using $\mathcal{T}$.

To formally define the notion of *best cover* we need to be able to characterize the notion of *"extra information"*, i.e., the information contained in one description and not contained in the other. For that, we use a non standard operation in description logics, the *difference or subtraction* operation. Roughly spoken, the difference of two descriptions is defined as being a description containing all information which is a part of one argument but not a part of the other one [TEE 94].

We formally define the *best covering* problem in a restricted framework of description logics where the difference operation is always semantically unique. We show that, in this framework, the problem of computing the *best covers* of a query Q using an ontology $\mathcal{T}$ can be seen as a new instance of the problem of rewriting concepts using a terminology [BEE 97, BAA 00]. A study of complexity showed that this problem is NP-Hard. Then, we make use of hypergraph theory to propose an algorithm that allows to compute the *best covers* of a concept Q using an ontology $\mathcal{T}$.

Context of this work The work presented in this paper has been developed and experienced in the context of the MKBEEM[3] project which aims at providing electronic marketplaces with intelligent, knowledge-based multilingual services. In this project, e-services are used to describe the offers delivered by the MKBEEM platform independently from specific providers. The reasoning mechanism described in this paper is used to allow clients to dynamically discover the available e-services that best meet their needs, to examine their properties and capabilities, possibly to provide missing information and to determine how to access them. First experimental results show that the modularity of the proposed architecture together with the associated reasoning mechanism allow to make the whole system provider-independent and more capable to face the great instability and the little lifetime of e-commerce offers and e-services.

3. MKBEEM stands for Multilingual Knowledge Based European Electronic Marketplace (IST-1999-10589, 1st Feb. 2000 - 1st Aug. 2002).

Organization of the paper The rest of this paper is organized as follows. Section 2 presents our motivation in using reasoning mechanisms for dynamic discovery of e-services. Section 3 introduces the description logic material and the reasoning mechanisms that are used in our framework. In Section 4, we provide a formal framework for the best covering problem and a way to solve it with the help of hypergraphs theory. In Section 5, we describe an algorithm we have implemented to compute the best covers of a concept Q using an ontology $\mathcal{T}$. Section 6 presents a brief description of the MKBEEM project. Section 7 reviews related work and presents future research directions.

2. Motivation

The main advantage of the proposed approach is to ground the dynamic discovery of e-services on a semantic comparison between a client query and available e-services. More precisely, we propose an algorithm that enables to achieve this semantic comparison by giving a way to extract from the e-services definitions the part that is semantically common with the query and the part that is semantically different from the query. Knowing the former and the latter allows to select relevant e-services and then to choose the best ones: this is the dynamic discovery. But knowing the latter allows also to initiate the dialogue between the e-commerce platform and the user in order to make him clarify his query.

The following example illustrates the practical interest of the reasoning mechanism described in this paper for our application.

Example 1

Let us consider an ontology[4] that contains the following e-services:
- $ToTravel$ allowing to consult a list of trips given a departure place, an arrival place, an arrival date and an arrival time,
- $FromTravel$ allowing to consult a list of trips given a departure place, an arrival place, a departure date and a departure time,
- $Hotel$ allowing to consult a list of hotels given a destination place, the check-in date, the check-out date, the number of adults and the number of children.

Now, assume we have the following query "I want to go from Paris to Madrid on Friday 21st of June, look for an accommodation there for one week (from 21st of June to 28th of June) and rent a car". Formally, the e-services $ToTravel$, $FromTravel$ and $Hotel$ as well as the query Q can be expressed as concept descriptions in a given description logic. Our goal is to rewrite Q into the closest description E expressed as a conjunction of e-services. Considering our ontology of e-services, the possibly interesting combinations of e-services are: $E_1 = \{Hotel, ToTravel\}$ and $E_2 = \{Hotel, FromTravel\}$. The two types of extra information brought by each

4. This ontology describes some e-services extracted from the French railways company (SNCF) web site (http://www.sncf.com).

Solution	Rest	Missing information
E_1	car rental, departure date	arrival date, arrival time, number of adults, number of children
E_2	car rental	departure time, number of adults, number of children

Table 1. *Example of extra information.*

combination of e-services are given in Table 1. For each combination, these two kinds of *"extra information"* are:

– the information which is contained in the query Q and not contained in its rewriting (cf. Table 1, column Rest), and

– the information contained in the rewriting and not contained in the query Q (cf. Table 1, column Missing information).

Continuing with the example, the best combinations are discovered by searching the ones that bring the least possible of extra information with respect to the query. It is clear that to better meet the user needs, it is more interesting to try to minimize, in first, the first kind of extra information (i.e., the column Rest). Here, the extra information of $ToTravel$ is "bigger" than the extra information of $FromTravel$. So, the best combinations for the query is $\{Hotel, FromTravel\}$. Once the best combinations have been dynamically discovered, a dialogue phase can be initiated with the user to ask him to provide the missing information.

3. Description Logics

The technical background of our proposal is constituted by Description Logics (DLs) enriched with a difference operator. We refer to [DON 96] for an introduction to DLs and to [TEE 94] for an extension of DLs with a difference operation. In this section, we introduce Description Logics, the difference operation (as defined by Teege in [TEE 94]) and the notion of size of a description.

3.1. *Description Logics: main notions*

DLs are a family of logics that were developed for modeling complex hierarchical structures and to provide a specialized reasoning engine to do inferences on these structures. The main reasoning mechanisms (like subsumption or satisfiability) are effectively decidable for some description logics ([DON 96]). Recently, DLs have been proved well-suited for the semantic Web. Some ontology languages such as OIL [FEN 01, HOR 02a] or DAML [HEN 00, HOR 02a] that were proposed to extend RDFS[5] are in fact syntactical variants of a very expressive DL.

5. Resource Description Framework Schema (http://www.w3.org/RDF/).

A DL allows to represent domain of interest in terms of *concepts* (unary predicates) that characterize subsets of the objects (*individuals*) in the domain, and *roles* (binary predicates) over such domain. Concepts are denoted by expressions formed by means of special constructors. Examples of constructors considered in this work are:

– the symbol $\top$ is a concept description which denotes the top concept while the symbol $\bot$ stands for the inconsistent (bottom) concept,

– concept conjunction ($\sqcap$), e.g., the concept description $parent \sqcap male$ denotes the class 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 constructors $(\geq n\ R)$ and $(\leq n\ R)$, e.g., the description $(\geq 1\, child)$ denotes the class of parents (i.e., individuals having at least one children), while the description $(\leq 1\ Leader)$ denotes the class of individuals that cannot have more than one leader.

The various description logics differ from one to another based on the set of constructors they allow. Table 2 below shows the constructors of two DLs: $\mathcal{FL}_0$ and $\mathcal{ALN}$. A concept obtained using the constructors of a description logic $\mathcal{L}$ is called an $\mathcal{L}$-concept. The semantics of a concept description is defined in terms of an interpreta-

Constructor 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	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$	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

Table 2. *Syntax and semantics of some concept-forming constructors.*

tion $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a nonempty set $\Delta^{\mathcal{I}}$, the domain of the interpretation, and an interpretation function $\cdot^{\mathcal{I}}$, which associates to each concept name $P \in \mathcal{C}$ a subset $P^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$ and to 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 2. Based on this semantics, subsumption, equivalence and the notion of least common subsumer[6] are defined as follows. Let $C_1, \ldots, C_n$ and D be concept descriptions:
• C is *subsumed by* D (noted $C \sqsubseteq D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.
• C is *equivalent to* D (noted $C \equiv D$) iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for all interpretation $\mathcal{I}$.

6. Informally, a least common subsumer of a set of concepts corresponds to the most specific description which subsumes all the given concepts [BAA 99].

• D is a least common subsumer of $C_1, \ldots, C_n$ (noted $D = lcs(C_1, \ldots, C_n)$) 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 $D \sqsubseteq D'$ [BAA 99].

The *intensional* component of a knowledge base built using a description logic 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 in the left-hand side of a definition. The concept name A is a defined concept in the terminology $\mathcal{T}$ iff it occurs in the left-hand side of a concept definition in $\mathcal{T}$.

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 constructors of a language $\mathcal{L}$ is called an $\mathcal{L}$-terminology. In the sequel, we assume that a terminology $\mathcal{T}$ is *acyclic*, i.e., there do not exist cyclic dependencies between concept definitions. 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 lcs of a set of descriptions can be obviously extended to concepts containing defined names. In this case we write $lcs_{\mathcal{T}}(C, D)$ to denote the least common subsumer of the concepts C and D w.r.t. a terminology $\mathcal{T}$ (i.e., the lcs is applied to the unfolded descriptions of C and D). In our application, an ontology of e-services will be described as a terminology (i.e., concept definitions are used to specify e-services). So, in the following, when appropriate, we use the term e-services (or simply services) to understand defined concepts in our application. Also, we use the terms terminology and ontology interchangeably.

Example 2

The e-services introduced informally in example 1 can be described using the description logic $\mathcal{FL}_0 \cup \{\geq\ n\ R\}$[7] as given in Table 3.
In the same way, the query given in example 1 could be abstracted by the following description:

$Q \doteq$ ($\geq$ 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

7. We note $\mathcal{FL}_0 \cup (\geq n\, R)$ the description logic $\mathcal{FL}_0$ augmented with the constructor $(\geq n\, R)$.

ToTravel	$\doteq$	($\geq$ 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)
FromTravel	$\doteq$	($\geq$ 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	$\doteq$	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)

Table 3. *Example of an ontology of e-services.*

3.2. *The difference operation*

In this section, we recall the main results obtained by Teege in [TEE 94] about 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 $C - D := \underset{\sqsupseteq}{max}\{B|B \sqcap D \equiv C\}$

This definition of difference requires that the second argument subsumes the first one. However, the difference $C - D$ between two incomparable descriptions C and 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 description logics, the set $C - D$ may contain descriptions which are not semantically equivalent as illustrated by the example below.

Example 3

Let us consider the following descriptions $C \doteq (\forall R.P) \sqcap (\forall R.\neg P)$ and $D \doteq (\forall R.P') \sqcap (\forall R (\leq 4 S))$. The following two non-equivalent descriptions $(\forall R.\neg P')$ and $(\forall R (\geq 5 S))$ are both members of the set $C - D$.

Teege [TEE 94] provides sufficient conditions to characterize the logics where the difference operation is always semantically unique and can be implemented in a simple syntactical way by constructing the set difference of subterms in a conjunction. Some basic notions and useful 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 $\{A_1, \ldots, A_n\}$.

– A clause set $A = \{A_1, \ldots, A_n\}$ is called **reduced** 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*, denoted by $\setminus_{\equiv}$, is defined as being the set difference of clause sets where clauses are compared on the basis of the equivalence relation. [TEE 94] provides two interesting results: *1)* in description logics with structurally unique RCFs, the difference operation can be straightforwardly calculated using the *structural difference operation*, and *2) structural subsumption* is a sufficient condition for a description logic to have structurally unique RCFs.

Consequently, structural subsumption is a sufficient condition that allows 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 [TEE 94] is different from the one usually used in the literature. Unfortunately, a consequence of this remark is that many description logics for which a structural subsumption algorithm exists (e.g., $\mathcal{ALN}$ [MOL 98]) do not have structurally unique RCFs. Nevertheless, the result given in [TEE 94] is still interesting in practice since there exists many description logics with this property. Examples of such logics include the language $\mathcal{FL}_0 \cup (\ge n\ R)$, that we have used in the context of the MKBEEM project, or the more powerful description logic $\mathcal{L}_1$ [TEE 94] , which contains the following constructors:

– $\sqcap, \sqcup, \top, \bot, (\ge n\ R)$, 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 $(\bot)$, composition $(\circ)$, differentiation $(|)$ for roles,

– bottom $(\bot)$ and composition $(\circ)$ for features.

In the rest of this paper we use the term *structural subsumption* in the sense of [TEE 94].

3.3. *Size of a description*

Let $\mathcal{L}$ be a description logic with structural subsumption. We define the size $|C|$ of an $\mathcal{L}$-concept description C as being the number of clauses in its RCFs[8]. If necessary,

8. We recall that, since $\mathcal{L}$ have structurally unique RCFs, all the RCFs of an $\mathcal{L}$-description are equivalent and thus have the same number of clauses.

a more precise measure of a size of a description can be defined by also 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 abstain from different descriptions of equivalent clauses. Please note that, in a description logic with structurally unique RCFs it is often possible to define a canonical form which is itself an RCF [TEE 94].

4. The best covering problem

In this section, we first investigate the *best covering* problem in the framework of description logics with structural subsumption. Then we see how to compute best covers using hypergraphs theory. Finally, we consider the running example to see what are the best covers of the query Q.

4.1. *Problem statement*

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

Definition 4 (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 being any conjunction of e-services occurring in $\mathcal{T}$ which shares some common information with Q. Please note that a cover E of Q is always consistent with Q (i.e., $Q \sqcap E \not\equiv \bot$) since $\mathcal{L}$ is a description logic with structurally unique RCFs[9] and we have $Q \not\equiv \bot$ and $S_i \not\equiv \bot, \forall i \in [1,n]$.

To define the notion of *best cover*, we first need to characterize more precisely the remaining descriptions both in the input concept description Q (hereafter called the *rest*) and in its cover E (hereafter called the *miss*).

Definition 5 (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, written $Rest_E(Q)$, is defined as follows: $Rest_E(Q) \doteq Q - lcs_{\mathcal{T}}(Q,E)$.

9. If the language $\mathcal{L}$ contains the incoherent concept $\bot$, then $\bot$ must be a clause, i.e., non trivial decompositions of $\bot$ is not possible (that means we cannot have incoherent conjunction of coherent clauses), otherwise it is easy to show that $\mathcal{L}$ does not have structurally unique RCFs.

The missing information of Q with respect to E, written $Miss_E(Q)$, is defined as follows: $Miss_E(Q) \doteq E - lcs_{\mathcal{T}}(Q, E)$.

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

Definition 6 (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

– there doesn't exist a cover E' of Q using $\mathcal{T}$ such that $(|Rest_{E'}(Q)|, |Miss_{E'}(Q)|) < (|Rest_E(Q)|, |Miss_E(Q)|)$, where $<$ stands for the lexicographic order.

The *best covering problem*, noted $\mathcal{BCOV}(\mathcal{T}, Q)$, 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 [BAA 00] (see [HAC 02] for a detailed proof).

4.2. *Computing best covers using hypergraphs*

Let us first recall some useful definitions regarding hypergraphs.

Definition 7 (hypergraph and transversals) [EIT 95]
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 an hypergraph $\mathcal{H}$ is noted $Tr(\mathcal{H})$.

Now we can show that the best covering problem can be interpreted in the framework of hypergraphs as the problem of finding the minimal transversals with a minimal cost.

Definition 8 (hypergraph $\mathcal{H}_{\mathcal{T}Q}$ generated from $\mathcal{T}$ and Q) Let $\mathcal{L}$ be a description logic with structural subsumption, $\mathcal{T}$ be an $\mathcal{L}$-terminology, and Q be an $\mathcal{L}$-concept description. Given an instance $\mathcal{BCOV}(\mathcal{T}, Q)$ of the best covering problem, we build an hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ as follows:

– each e-service S_i in $\mathcal{T}$ becomes 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 $i \in [1,k]$, becomes an edge in $\mathcal{H}_{\mathcal{T}Q}$, noted w_{A_i}, with $w_{A_i} = \{V_{S_i} \,|\, S_i \in \mathcal{S}_{\mathcal{T}}$ 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.

For the sake of clarity we introduce the following notation.

Notation For any set of vertices $X = \{V_{S_i}\}$, subset of Σ, we note $E_X \doteq \sqcap_{V_{S_i} \in X} S_i$ the concept obtained from the conjunction of the e-services corresponding to the vertices in X. Mutually, for any concept $E \doteq \sqcap_{j \in [1,m]} S_{i_j}$, we note $X_E = \{V_{S_{i_j}}, j \in [1,m]\}$ the set of vertices corresponding to the e-services in E.

With lemmas 1 and 2 given below, we show 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 non empty edges. Proofs of these lemmas are in [HAC 02].

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

Lemma 2 (characterization of covers that minimize the rest) Let $\widehat{\mathcal{H}}_{\mathcal{T}Q} = (\Sigma, \Gamma')$ be the hypergraph built by removing from $\mathcal{H}_{\mathcal{T}Q}$ the empty edges. A rewriting $E_{min} \doteq S_{i_1} \sqcap \ldots \sqcap S_{i_m}$, 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 $Rest_{E_{min}}(Q)$ iff $X_{E_{min}} = \{V_{S_{i_j}}, j \in [1,m]\}$ is a transversal of $\widehat{\mathcal{H}}_{\mathcal{T}Q}$.

Having covers that minimize the rest, it remains to isolate those minimizing the miss in order to have the best covers. To express miss minimization in the hypergraphs framework, we introduce the following notion of cost.

Definition 9 (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 $\mathcal{BCOV}(\mathcal{T}, Q)$ problem can be reduced to the computation of the transversals with minimal cost of the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$. Clearly, it appears that we can only care about *minimal* transversals. To sum up, 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, one can reuse results known for computing minimal transversals for solving the best covering problem.

Example 4

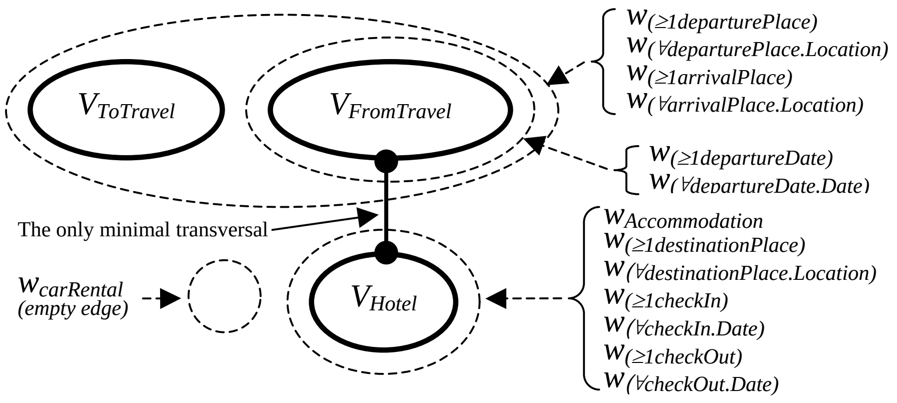
Let $\mathcal{T}$ and Q be, respectively, the e-services ontology and the query given in the example 2. We assume that the concept names (e.g., Location, Date, Accommodation, ...), that appear in the description of the query Q and/or in the descriptions of the e-services of $\mathcal{T}$, are all atomic concepts. Hence, the query Q and the e-services of $\mathcal{T}$ are all provided by their RCFs[10]. Therefore, the associated hypergraph $\mathcal{H}_{\mathcal{T}Q} = (\Sigma, \Gamma)$ will be made of the set of vertices $\Sigma = \{V_{ToTravel}, V_{FromTravel}, V_{Hotel}\}$ and the set Γ containing the following edges:

Edges created from Q's clauses		Set of e-services/vertices
$w_{(\geq 1 departurePlace)}$	$=$	$\{V_{ToTravel}, V_{FromTravel}\}$
$w_{(\forall departurePlace.Location)}$	$=$	$\{V_{ToTravel}, V_{FromTravel}\}$
$w_{(\geq 1 arrivalPlace)}$	$=$	$\{V_{ToTravel}, V_{FromTravel}\}$
$w_{(\forall arrivalPlace.Location)}$	$=$	$\{V_{ToTravel}, V_{FromTravel}\}$
$w_{(\geq 1 departureDate)}$	$=$	$\{V_{FromTravel}\}$
$w_{(\forall departureDate.Date)}$	$=$	$\{V_{FromTravel}\}$
$w_{Accommodation}$	$=$	$\{V_{Hotel}\}$
$w_{(\geq 1 destinationPlace)}$	$=$	$\{V_{Hotel}\}$
$w_{(\forall destinationPlace.Location)}$	$=$	$\{V_{Hotel}\}$
$w_{(\geq 1 checkIn)}$	$=$	$\{V_{Hotel}\}$
$w_{(\forall checkIn.Date)}$	$=$	$\{V_{Hotel}\}$
$w_{(\geq 1 checkOut)}$	$=$	$\{V_{Hotel}\}$
$w_{(\forall checkOut.Date)}$	$=$	$\{V_{Hotel}\}$
$w_{carRental}$	$=$	$\emptyset$

We can see that no e-service 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: $Rest_{min} \equiv carRental$ (cf lemma 1). Now, considering the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$, the only minimal transversal is: $X = \{V_{FromTravel}, V_{Hotel}\}$. So, $E_X \doteq Hotel \sqcap FromTravel$ is the best cover of Q using the ontology of e-services $\mathcal{T}$. Figure 1 shows the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ and its only minimal transversal which corresponds to the only best cover of Q.

If there were many minimal transversals, we would compute their cost, that is the size of the missing information of their corresponding description. For example, the size of the missing information of E_X is the cost of the transversal X which is given below.
$cost(X) = |Miss_{E_X}(Q)| = |Miss_{FromTravel \sqcap Hotel}(Q)|$
$cost(X) = |$($\geq$ 1 departureTime) $\sqcap$ ($\forall$ departureTime.Time) $\sqcap$ ($\geq$ 1 nbAdults) $\sqcap$ ($\forall$ nbAdults.Integer) $\sqcap$ ($\geq$ 1 nbChildren) $\sqcap$ ($\forall$ nbChildren.Integer)$| = 6$.
And then the best covers would be the minimal transversals with the minimal cost. In this example, we do not care about this cost because the hypergraph $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ has only one minimal transversal.

10. Otherwise, we have to recursively unfold the e-service (resp. query) description by replacing by its definition each concept name appearing in the e-service (resp. query) description.



$V_{ToTravel}$
$V_{FromTravel}$
V_{Hotel}
$w_{(\geq 1 departurePlace)}$
$w_{(\forall departurePlace.Location)}$
$w_{(\geq 1 arrivalPlace)}$
$w_{(\forall arrivalPlace.Location)}$
$w_{(\geq 1 departureDate)}$
$w_{(\forall departureDate.Date)}$
The only minimal transversal
$w_{Accommodation}$
$w_{(\geq 1 destinationPlace)}$
$w_{(\forall destinationPlace.Location)}$
$w_{(\geq 1 checkIn)}$
$w_{(\forall checkIn.Date)}$
$w_{(\geq 1 checkOut)}$
$w_{(\forall checkOut.Date)}$
$w_{carRental}$ (empty edge)

Figure 1. $\widehat{\mathcal{H}}_{\mathcal{T}Q}$ *and its only minimal transversal.*

5. Algorithm

In this section we give a sketch of an algorithm, called *computeBCov*, for computing the best covers of a concept Q using a terminology $\mathcal{T}$. In the previous section , we have shown 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 an hypergraph is central in various fields of computer science [EIT 95]. The precise complexity of this problem is still an open problem. In [MIC 96], 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 our knowledge, this is the best theoretical time bound for the problem of the generation of the transversal hypergraph.

In our case, since the problem is slightly different, we propose an adaptation of an existing algorithm with a combinatorial optimization technique (branch-and-bound) to compute the transversals with a minimum cost.

A classical algorithm for computing the minimal transversals of an hypergraph is presented in [BER 89, MAN 94, EIT 95]. 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 between the candidates generated in the previous step and each vertex in the considered edge. At each step, the non-minimal candidate transversals are pruned.

So, a naive approach to compute the minimal transversals with a minimal cost would be to compute all the minimal transversals, using such an algorithm, and then to choose those transversals which have the minimal cost. The algorithm *computeBCov* presented here makes an improvement over the naive approach by using an additional pruning criteria for reducing the number of candidates in the intermediate steps of such a classical algorithm, while ensuring that the transversals with the minimal cost are still considered. The main idea behind this algorithm is to use a Branch-and-

Algorithm 1 ***computeBCov*** (sketch)

```
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 new generated set of the candidate transversals.
 6:    Remove from $Tr$ the transversals which are non minimal and those whose cost
       is 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 $E_X$.
11: end for
```

Bound like enumeration of transversals. First, a simple heuristic is used to efficiently compute a cost of a *good* transversal (i.e., a transversal expected to have a small cost) (line 3). 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 $CostEval$. As we have, for any set of vertices $X = \{S_i\}$:

$$cost(X) = |Miss_{E_X}(Q)| \leq \sum_i |Miss_{S_i}(Q)| = \sum_{S_i \in X} cost(\{S_i\})$$

the evaluation is an upper bound of the cost of a feasible transversal. Then as we consider candidates in intermediate steps of the algorithm, we can eliminate from Tr any candidate transversal that has a greater cost than $CostEval$, since that candidate could not possibly lead to a transversal that is better than what we already know (line 6). Then, from each candidate transversal that remains in Tr, 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 translated into a concept E_X which constitutes an element of the solution to the $\mathcal{BCOV}(\mathcal{T}, Q)$ problem.

6. Experimentation: The MKBEEM project

The work presented in this paper has been developed and used in the context of the MKBEEM project which aims at providing electronic marketplaces with intelligent, knowledge-based multilingual services [mkb]. In this project, ontologies are used to provide a consensual representation of the electronic commerce field in two typical Domains (Tourism and Mail order). The MKBEEM ontologies are structured in three layers, as shown in Figure 2.

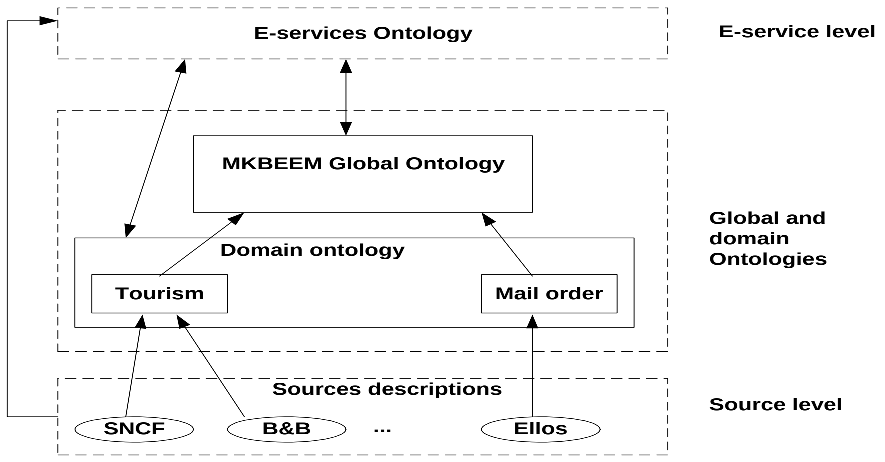


Figure 2. *Knowledge representation in the MKBEEM system.*

The global ontology describes the common terms used in the whole MKBEEM platform while each domain ontology contains specific concepts corresponding to one of the domains of the MKBEEM partners (e.g, tourism, mail orders, etc.). The sources descriptions specify the providers competencies, i.e., the description of the contents of the providers information sources. Finally, all the offers available in the MKBEEM platform are integrated and described in the e-services ontology. Whereas in many e-commerce platforms e-services are associated to providers, we have defined an e-service as a *provider-independent offer* available on a given e-commerce platform. The example 1 given in section 2, is a typical mediation instance in the context of this project: the user poses queries in terms of the “integrated schema" (i.e., e-services and domain ontology) rather than directly querying specific provider information sources. This enables users to focus on *what* they want, rather than worrying about *how* and *from where* to obtain the answers. Then, to effectively handle mediation tasks, the MKBEEM system rely on two reasoning mechanisms:

– the first allows to reformulate users queries against the domain ontology in terms of e-services. The aim here is to allow the users/applications to automatically discover the available e-services that best meet their needs, to examine their capabilities and possibly to complete missing information;

– the second, called *query plan generation*, takes place after the first step and allows to reformulate a user query, expressed as a combination of e-services, in terms of providers views. The aim of this second issue is to allow the identification of the views that are able to answer to the query (knowing that, afterwards, the query plans will be translated into databases queries via the corresponding wrappers).

While the second reasoning mechanism, known as *query rewriting using views*, has already been addressed in the literature [BEE 97, GOA 00], the first is a new problem for which we have proposed a solution in this paper.

The algorithm *computeBCov* presented in section 5 has been implemented as an integrated component in the MKBEEM prototype. This prototype is built as a set of Enterprise Java Beans (EJB) components that interact with each other. The prototype relies on the Picsel [GOA 00] mediator to handle the *query plan generation* task. There are also some components dedicated to the interaction with the user interface built with Java Server Pages (JSPs) or Servlets. Finally, some of these components have the functionality of interacting with the remote (or locally duplicated) databases in the provider information systems.

The MKBEEM prototype has been validated on a pan-European scale (France and Finland), with three basic languages (Finnish, English and French) and two optional languages (Spanish and Swedish), in two distinct end-user fields: 1) Business to consumer on-line sales, and 2) Web based travel/tourism services. In our first experiments we used small ontologies ($\simeq 500$ concepts and 50 e-services) to validate the accuracy of the suggested approach. On-going work is devoted to the assessment of the performance and the scalability of the MKBEEM prototype.

7. Discussion

Existing solutions that achieve dynamic discovery of e-services rely on simple query mechanisms to provide *individual* services that *exactly* match the query. Clearly, a semantic match like the one proposed in this paper is beyong the representation capabilities of the emerging XML based standards and current e-service platforms. For example, the information provided in a UDDI business registration consists of three components: "white pages" (e.g., business name, contact information, ...); "yellow pages" including industrial categorizations based on standard taxonomies; and "green pages", the technical information about services that are exposed by the business. Based on these descriptions, UDDI provides poor search facilities allowing only a keyword based search of businesses, services and the so-called TModels on the bases of their names.

To cope with these limitations, there are some proposals of matching algorithms that employ semantic web technology for service description [GON 01, PAO 02]. [GON 01] reports on an experience in building matchmaking prototype based on description logic reasoner and operating on service descriptions in DAML+OIL [HOR 02b]. The proposed matching algorithm is based on simple subsumption and consistency tests. [PAO 02] proposes a more elaborated matching algorithm between services and requests described in DAML-S[11]. The algorithm recognizes various degrees of

11. http://www.daml.org/services/

matching that are determined by the minimal distance between concepts in the concept taxonomy. The problem of capabilities based matching has also been addressed by the multi-agent community. A matching algorithm is proposed in [SYC 02] for the language LARKS. This algorithm is similar to the one proposed in [PAO 02] since LARKS identifies a set of filters that progressively restrict the number of services that are candidates for a match. Our work falls in this research stream of approaches that support the location of e-services based on a semantic match between declarative descriptions of services and requests. However, since we view the e-service discovery as a rewriting process, our algorithm is able to discover *combinations* of services that match (cover) a given query. Furthermore, the difference between the query and its rewriting (i.e., rest and miss) is effectively computed and can be used to improve the e-service interoperability. So our dynamical discovery of e-services appears as the first step towards a dynamic composition of e-services.

From the theoretical point of view, the best covering problem belongs to the general framework for *rewriting using terminologies* provided in [BAA 00]. 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 in order to select the relevant rewritings. Already investigated instances of this problem are the minimal rewriting problem [BAA 00] and rewriting queries using views [BEE 97, GOA 00]. In the former, ρ is instantiated by equivalence modulo $\mathcal{T}$ and the size of the rewriting is used as the optimality criterion. In the latter, which is the problem underlying the query plan generation in MKBEEM, the relation ρ is instanciated by subsumption and the optimality criterion is the inverse subsumption [BAA 00]. In this context ,the *best covering problem* is the new instance of the problem of rewriting concepts using terminologies where the goal is to rewrite a description Q into the closest description expressed as a conjunction of (some) concept names in $\mathcal{T}$ (hence, ρ is neither equivalence nor subsumption).

We have investigated this problem in a 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. This framework appears to be sufficient in the context of the MKBEEM project. But the languages that are recommended to realize the semantic web vision tend to be more expressive. That's why 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 (for example $\mathcal{ALN}$) 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. However, after the very first results we got concerning $\mathcal{ALN}$, we argue that a restricted difference operator can be defined, and then the framework can be extended, so that many practical applications of the dynamic discovery of e-services can be solved with this more expressive logic.

8. References

[BAA 99] BAADER F., KÜSTERS R., MOLITOR R., “Computing Least Common Subsumer in Description Logics with Existential Restrictions”, DEAN T., Ed., *Proc. of the* 16^{th} *Int. Joint Conf. on AI*, M.K, 1999, p. 96-101.

[BAA 00] BAADER F., KÜSTERS R., MOLITOR R., “Rewriting Concepts Using Terminologies”, *Proc. of the Int. Conf. KR Colorado, USA*, Apr. 2000, p. 297-308.

[BEE 97] BEERI C., LEVY A., ROUSSET M.-C., “Rewriting Queries Using Views in Description Logics”, YUAN L., Ed., *Proc. of the* ACM PODS*, New York, USA*, Apr. 1997, p. 99-108.

[BEN 02] BENATALLAH B., DUMAS M., SHENG Q., NGU A., “Declarative Composition and Peer-to-Peer Provisioning of Dynamic Web Services”, *Proc. of the IEEE Int. Conf. on Data Engineering , San Jose, USA*, Jun. 2002, p. 297–308.

[BER 89] BERGE C., *Hypergraphs*, vol. 45 of *North Holland Mathematical Library*, Elsevier Science Publishers B.V. (North-Holland), 1989.

[CAS 01a] CASATI F., SHAN M.-C., “Dynamic and adaptive composition of e-services”, *Information Systems*, vol. 26, num. 3, 2001, p. 143-163.

[CAS 01b] CASATI F., SHAN M.-C., “Models and Languages for Describing and Discovering E-Services”, *Proceedings of SIGMOD 2001, Santa Barbara, USA*, May 2001.

[DON 96] DONINI F., M. LENZERINI D. NARDI A. S., “Reasoning in description logics”, *In Gerhard Brewka, editor, Foundation of Knowledge Representation*, CSLI-Publications, 1996, p. 191-236.

[EIT 95] EITER T., GOTTLOB G., “Identifying the Minimal Transversals of a hypergraph and Related Problems”, *SIAM Journal on Computing*, vol. 24, num. 6, 1995, p. 1278–1304.

[FEN 01] FENSEL D., VAN HARMELEN F., HORROCKS I., MCGUINNESS D., PATEL-SCHNEIDER P. F., “OIL: An ontology infrastructure for the semantic web.”, *IEEE Intelligent Systems*, vol. 16, num. 2, 2001, p. 38–45.

[FEN 02] FENSEL D., BUSSLER C., MAEDCHE A., “Semantic Web Enabled Web Services”, *International Semantic Web Conference, Sardinia, Italy*, Jun. 2002, p. 1–2.

[GOA 00] GOASDOUÉ F., V. LATTÈS M.-C. R., “The Use of CARIN Language and Algorithms for Information Integration: The PICSEL System”, *IJICIS*, vol. 9, num. 4, 2000, p. 383-401.

[GON 01] GONZÁLEZ-CASTILLO J., TRASTOUR D., BARTOLINI C., “Description Logics for Matchmaking of Services”, *Proc. of the KI-2001 Workshop on Applications of Description Logics Vienna, Austria*, vol. 44, September 2001.

[HAC 02] HACID M.-S., LÉGER A., REY C., TOUMANI F., “Dynamic discovery of e-services: a description logics based approach.”, Report, 2002, LIMOS, Clemont-Ferrand, France, see http://lisi.insa-lyon.fr/∼mshacid/publications.html.

[HEN 00] HENDLER J., MCGUINNESS D. L., “The DARPA Agent Markup Language”, *IEEE Intelligent Systems*, vol. 15, num. 6, 2000, p. 67–73.

[HOR 02a] HORROCKS I., P.F.PATEL-SCHNEIDER, VAN HARMELEN F. ., “Reviewing the Design of DAML+OIL: An Ontology Language for the Semantic Web”, *Proc. of the 18th Nat. Conf. on Artificial Intelligence (AAAI 2002)*, 2002, To appear.

[HOR 02b] HORROCKS I., “DAML+OIL: A Reason-able Web Ontology Language”, *Proc. of the Int. Conf. on Extending Database Technology Prague, Czech Republic*, Mar. 2002, p. 2–13.

[MAN 94] MANNILA H., RäIHä K.-J., *The Design of Relational Databases*, Addison-Wesley, Wokingham, England, 1994.

[MIC 96] MICHAEL L. FREDMAN L. K., “On the Complexity of Dualization of Monotone Disjunctive Normal Forms.”, *Journal of Algorithms*, vol. 21, num. 3, 1996, p. 618-628.

[mkb] “MKBEEM web site : http://www.mkbeem.com”.

[MOL 98] MOLITOR R., “Structural subsumption for $\mathcal{ALN}$”, report num. LTCS-98-03, March 1998, Aachen University of Technology, Research Group for Theoretical Computer Science.

[PAO 02] PAOLUCCI M., KAWAMURA T., PAYNE T., SYCARA K., “Semantic Matching of Web Services Capabilities.”, *Proc. of the Int. Semantic Web Conference, Sardinia, Italy*, June 2002, p. 333–347.

[SYC 02] SYCARA K., WIDOFF S., KLUSCH M., , LU J., “LARKS: Dynamic Matchmaking Among Heterogeneous Software Agents in Cyberspace”, *Autonomous Agents and Multi-Agent Systems*, vol. 5, 2002, p. 173–203.

[TEE 94] TEEGE G., “Making the Difference: A Subtraction Operation for Description Logics”, DOYLE J., SANDEWALL E., TORASSO P., Eds., *Proc. of the Int. Conf. KR*, San Francisco, CA, 1994, Morgan Kaufmann.

[Vld01] “The VLDB Journal: Special Issue on E-Services”, 10(1), Springer-Verlag Berlin Heidelberg, 2001.

[Wei01] “Data Engineering Bulletin: Special Issue on Infrastructure for Advanced E-Services”, 24(1), IEEE Computer Society, 2001.