Finding Informative Commonalities in Concept Collections

Simona Colucci
Politecnico di Bari
Via Orabona 4
70125, Bari, Italy
s.colucci@poliba.it

Eugenio Di Sciascio
Politecnico di Bari
Via Orabona 4
70125, Bari, Italy
disciascio@poliba.it

Francesco M. Donini
Università della Tuscia
Via San Carlo 32
01100, Viterbo, Italy
donini@unitus.it

Eufemia Tinelli
Università di Bari
Via Orabona 4
70125, Bari, Italy
tinelli@di.uniba.it

ABSTRACT

The problem of finding commonalities characterizes several Knowledge Management scenarios involving collection of resources. The automatic extraction of shared features in a collection of resource descriptions formalized in accordance with a logic language has been in fact widely investigated in the past. In particular, with reference to Description Logics concept descriptions, Least Common Subsumers have been specifically introduced.

Nevertheless, such studies focused on identifying features shared by the whole collection. The paper proposes instead novel reasoning services in Description Logics, aimed at identifying commonalities in a significant portion of the collection, rather than in the collection as a whole.

In particular, common subsumers adding informative content to the one provided by the Least Common Subsumer are here investigated.

The new services are useful in all scenarios where features are not required to be fully shared, like the one motivating our research: Core Competence extraction in knowledge intensive companies.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation Languages; H.3.3 [**Information Search and Retrieval**]: Clustering

General Terms

Algorithms, Management

Keywords

Description Logics, Non-standard Inferences, Informative Common Subsumers

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.

1. INTRODUCTION

Every scenario characterized by the presence of a collection of resources faces the problem of finding common features in the collection, regardless of the nature of involved resources. If collected elements are described according to a formal language conveying semantics, the process of identifying such commonalities can be automatically performed. In particular, Description Logics [1] offer inference services specifically aimed at identifying concept collection commonalities. Least Common Subsumers(LCS) have in fact been defined [9] — originally for the DL underlying Classic [8]— with the specific purpose of determining the most specific concept description subsuming all of the elements of a given collection.

Usefulness of LCSs has been shown in several different application fields. As an example, in the bottom-up construction of knowledge bases [3], the knowledge engineer may define several typical objects of the domain, which are then generalized through the computation of their LCS. Such an approach overcomes the drawbacks of classical top-down techniques, in which the general concept descriptions are defined first.

Inductive learning algorithms also employ LCS computation to find a least general concept consistent with a set of positive example, used as basis for learning [10].

The measure of concept similarity in specific information retrieval [20] represents one more application of LCS.

Noteworthy, all above introduced application scenarios share the need of individuating features which are common to all of the elements in a given collection.

In other applications, instead, the issue is determining commonalities of a significant portion of the collection rather than of the collection as a whole. The problem reverts then to finding a concept subsuming a significant number, or percentage, of elements in the collection. In an organizational scenario, for example, a crucial strategical issue consists in determining the know-how specializing the company. Such a knowledge takes the name "Core Competence" [15] in knowledge management literature and represents the fields of excellence for a company, *e.g.*, the competence to invest on in long term strategy. If employee profiles and organization knowledge are formalized in a collection of concept descriptions, the management could extract company Core Competence by searching for features shared by a significant num-

ber of employees. The degree of significance may be chosen by the management on the basis of organizational needs.

In order to perform such extraction process in a Knowledge Representation framework, we need to define concepts which are LCS of k elements in a collection of n descriptions, with $k < n$. We give the name **k-Common Subsumers** to such concepts.

As the LCS of the whole collection is by definition a k-Common Subsumer for every $1 < k < n$, we define distinguished **Informative k-Common Subsumers**, to denote k-Common Subsumers adding informative content to the LCS. k-Common Subsumers equivalent to the LCS would not justify in fact the introduction of a reasoning service other than the LCS.

Imagine now that the management of a company in the organizational scenario introduced before needs to extract the features common to the biggest number of employees in the company: we define to this aim **Best Common Subsumers**, which are concepts subsuming a number $\overline{k}$ of elements in the collection, where $\overline{k}$ is the biggest number of subsumed concepts.

If the collection admits an LCS not equivalent to the universal concept, the Best Common Subsumer is obviously the LCS itself. In this case we therefore define the **Best Informative Common Subsumers**, which are concepts subsuming the biggest number of elements in the collection less than the cardinality n.

The rest of this paper is organized as follows: in the next Section we briefly introduce the DL formalism we adopt; then we outline our motivational case study and its formalization in DLs. In Section 4 we detail all reasoning services needed for the commonalities extraction process. Solving algorithms are provided in Section 5 w.r.t. different DLs, together with some computational complexity results. A case study illustrating introduced services in the organizational context is presented in Section 6, before closing the paper with conclusions.

2. BASIC DESCRIPTION LOGICS

We start recalling here basics of the formalism we adopt, based on Description Logics (DLs). DLs are a family of formalisms and reasoning services widely employed for knowledge representation in a decidable fragment of First Order Logic.

The alphabet of each DL is therefore made up by unary and binary predicates, denoted as **Concepts Names** and **Role Names**, respectively. The domain of interest is represented through more expressive and complex **Concept Descriptions**, involving *constructors* over concept and role names. The set of constructors allowed by a DL characterizes it in terms of expressiveness and reasoning complexity: of course the more a DL is expressive, the harder is inferring new knowledge on its descriptions.

Concept Definitions allow to assign an unique concept name to complex concept descriptions: the so called **Unique Name Assumption** (UNA) holds in every DL. Names associated to concept descriptions constitute the set of *Defined Concepts*, distinguished by *Primitive Concepts* not appearing on the left hand side of any concept definition and corresponding to the concept names. Defined and primitive concepts constitute the set of concept names, which will be referred to by the letter A in the following; C and D will denote instead arbitrary concept descriptions.

Concept Inclusions detail instead specificity hierarchies among concepts, either defined or primitives.

Role Inclusions are also aimed at detailing specificity hierarchies, but among roles.

The set of concepts inclusions and definitions represents the formal representation of the domain of interest, the intensional knowledge which takes the name **TBox** in DL systems and **Ontology** in the generic knowledge representation framework. TBoxes containing recursive concept definitions are called *cyclic*(*acyclic* otherwise).

The semantic of concept descriptions is conveyed through an **Interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ is a non empty set denoting the domain of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is an interpretation function such that:

- $\cdot^{\mathcal{I}}$ maps each concept name A in a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
- $\cdot^{\mathcal{I}}$ maps each role name R in a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Possible DL constructors and the related semantics are shown in Table 1, which also shows the constructors allowed by the DLs investigated in this paper.

The behavior of the interpretation function in presence of definitions and inclusions is detailed in Table 2 and can be inductively explained by taking the semantics of allowed constructors into account. Table 2 also shows assertions allowed by all DLs investigated in the paper.

An interpretation I is a **model** for a TBox $\mathcal{T}$ if it satisfies the whole set of assertions in $\mathcal{T}$.

The motivating scenario we here propose needs at least the expressiveness of $\mathcal{ELHN}$ sublanguage of DLs for resources representation. The full expressiveness of $\mathcal{ELHN}$ is explained, with the aid of constructors usage examples, in Table 3.

Ontologies using DL can be easily modeled using languages for the Semantic Web [13, 19, 23]. These languages have been conceived to allow for representation of machine understandable, unambiguous, description of web content through the creation of domain ontologies, and aim at increasing openness and interoperability in the web environment. The strong relations between DLs and the above introduced languages for the Semantic Web [2] is also evident in the definition of the three OWL sub-languages:

OWL-Lite: allows class hierarchy and simple constraints on relation between classes;

OWL-DL: is based on Description Logics theoretical studies, it allows a great expressiveness keeping computational completeness and decidability;

OWL-Full: using such a language, there is a huge syntactic flexibility and expressiveness. This freedom is paid in terms of no computational guarantee.

3. MOTIVATING CASE STUDY

The investigations on the problem of finding informative commonalities in concept collections originate from a real need we faced in the implementation of IMPAKT, a novel and optimized semantic-based knowledge management system, which will be released late this year. IMPAKT is specifically aimed at semantic-based human resources management [11] and provides Core Competence extraction as decisional support service. IMPAKT ensures the scalability

Constructor Name	Syntax	Semantics	$\mathcal{ELHN}$	$\mathcal{ALN}$	$\mathcal{ALE}$	$\mathcal{ALEN}$
top-concept	$\top$	$\Delta^{\mathcal{I}}$	x	x	x	x
bottom-concept	$\bot$	$\emptyset$	x	x	x	x
atomic negation	$\neg A$	$\Delta^{\mathcal{I}} \backslash A^{\mathcal{I}}$	x	x	x	x
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	x	x	x	x
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$		x	x	x
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$	x	$\exists r.\top$	x	x
at-least restriction	$(\geq\ n\ r)$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \geq n\}$	x	x		x
at-most restriction	$(\leq\ m\ r)$	$\{x \in \Delta^{\mathcal{I}} \mid \sharp\{y \in \Delta^{\mathcal{I}} \mid (x,y) \in r^{\mathcal{I}}\} \leq m\}$	x	x		x

Table 1: DLs Set of Conctructors

	Syntax	Semantics	$\mathcal{ELHN}$	$\mathcal{ALN}$	$\mathcal{ALE}$	$\mathcal{ALEN}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$	x	x	x	x
concept inclusion	$A \sqsubseteq D$	$A^{\mathcal{I}} \subseteq D^{\mathcal{I}}$	x	x	x	x
role inclusion	$r \sqsubseteq q$	$r^{\mathcal{I}} \subseteq q^{\mathcal{I}}$	x			

Table 2: Syntax and Semantics of TBox assertions

Expression	Example	Explanation
top-concept		whole domain
bottom-concept		empty set
atomic negation	$\neg$`Skill`	concepts which are not skills
conjunction	`C++` $\sqcap$ `Java`	knowledge about both Java and C++
existential restriction	$\exists$`knowsLanguage.English`	elements of the domain knowing English as foreign language
at-least restriction	($\geq$ 3 `hasExperienceYears`)	elements of the domain endowed with more than 3 years of working experience
at-most restriction	($\leq$ 2 `knowsLanguage`)	elements of the domain knowing not more than two foreign language
concept definition	`Engineer` $\equiv$ $\exists$`hasMasterDegree.Engineering`	An engineer is an element of the domain endowed with a degree in engineering
concept inclusion	`Consultant` $\sqsubseteq$ $\neg$`Employee`	Elements of the domain working as consultants may not be included in employees set
role inclusion	`advancedKnowledge` $\sqsubseteq$ `basicKnowledge`	Elements of the domain endowed with a specific knowledge at an advanced level hold the same knowledge at a basic level

Table 3: $\mathcal{ELHN}$ expressiveness explained

of proposed knowledge management approaches by means of a smart pre-classification of semantic-based information.

In [15] the notion of *Core Competence* was introduced to indicate the strategic knowledge of a company. A core competence is defined as a sort of capability providing customer benefits, hard to be imitated from competitors and possessing leverage potential.

Further definitions of Core Competence have been proposed in the literature in the attempt of finding methods for detecting such a specializing knowledge [18, 22].

The process of individuating Core Competence is in fact usually characterized by high complexity and low objectivity because of the intangibility of knowledge itself and difficulties inherent in formalizing them.

The automation of Core Competence extraction process asks in fact for company know-how to be described according to a language endowed with formal semantics.

Hereafter we use Description Logics for knowledge representation and, for the sake of simplicity, assume that the only source of company know-how is company personnel. As a result, a company which needs to automatically extract its Core Competence has to formalize the knowledge profiles of employees according to the vocabulary provided by a TBox describing skill management domain.

An excerpt of the inclusions and the assertions composing the TBox at the basis of our case study is given in Figure 1 and Figure 2, respectively.

$$\texttt{Operations Management} \sqsupseteq \left\{ \begin{array}{l} \texttt{OperationsOptimization} \\ \texttt{Process Management} \\ \texttt{ProductionManagement} \end{array} \right.$$

$$\texttt{SoftwareEngineering} \sqsupseteq \{\ \texttt{UML}$$

$$\texttt{Programming} \sqsupseteq \left\{ \begin{array}{l} \texttt{ScriptLanguages} \sqsupseteq \left\{ \begin{array}{l} \texttt{Javascript} \\ \texttt{VBscript} \end{array} \right. \\ \texttt{OOP} \sqsupseteq \left\{ \begin{array}{l} \texttt{Java} \\ \texttt{C++} \end{array} \right. \\ \texttt{StructuralProgramming} \sqsupseteq \texttt{C} \end{array} \right.$$

$$\texttt{InformationSystems} \sqsupseteq \left\{ \begin{array}{l} \texttt{DBMS} \\ \texttt{ERPsystem} \sqsupseteq \{\ \texttt{SAP} \\ \texttt{TCP/IP} \end{array} \right.$$

$$\texttt{OperatingSystems} \sqsupseteq \left\{ \begin{array}{l} \texttt{Unix} \\ \texttt{Windows} \end{array} \right.$$

$$\texttt{AssetAllocation} \sqsupseteq \texttt{HumanResourcesManagement}$$

$$\texttt{basicKnowledge} \sqsupseteq \texttt{advancedKnowledge}$$

Figure 1: TBox Inclusions

The extraction process is grounded on the reasonable assumption that Core Competence, although characterizing a company, has not necessarily to be held by the whole personnel, but at least by a significant portion of it. A competence shared by all of the employees could be in fact too generic, if the objective is, for example, identifying skills to invest on in long term strategy.

Consider the tiny organizational scenario in which the following employees are employed:

- **Antonio:** Computer science engineer with advanced knowledge about Java since more than 3 years, UML since more than 3 years and about Unix and a basic knowledge of Information Systems;
- **Claudio:** Managerial Engineer with basic knowledge about SAP and software engineering and advanced knowledge about Java since more than 2 years;
- **Roberto:** Asset Manager with advanced knowledge about human resources management since more than 5 years and C++ and VB Script;
- **Daniele:** Engineer with basic knowledge of data base management systems and advanced knowledge of C since more than 5 years and Javascript since 3 years.

$$\texttt{Manager} \equiv \exists \texttt{advancedKnowledge.}(\texttt{ManagementTechniques} \sqcap (\geq\ 8\ \texttt{hasExperienceYears}))$$

$$\texttt{AssetManager} \equiv \texttt{Manager} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{AssetAllocation} \sqcap (\geq\ 5\ \texttt{hasExperienceYears}))$$

$$\texttt{Engineer} \equiv \exists \texttt{advancedKnowledge.}(\texttt{Design} \sqcap (\geq\ 5\ \texttt{hasExperienceYears})) \sqcap \exists \texttt{hasMasterDegree.Engineering} \sqcap \exists \texttt{basicKnowledge.OperationsOptimization}$$

$$\texttt{ManagerialEngineer} \equiv \texttt{Engineer} \sqcap \exists \texttt{basicKnowledge.ERPsystem} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{ProductionManagement} \sqcap (\geq\ 3\ \texttt{hasExperienceYears})) \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Process Management} \sqcap (\geq\ 3\ \texttt{hasExperienceYears}))$$

$$\texttt{CSEngineer} \equiv \texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.OperatingSystems} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap (\geq\ 5\ \texttt{hasExperienceYears})) \sqcap \exists \texttt{advancedKnowledge.}(\texttt{SoftwareEngineering} \sqcap (\geq\ 3\ \texttt{hasExperienceYears}))$$

Figure 2: TBox Definitions

The four profiles are formalized according to the given TBox as shown in Table 4. It is noteworthy that the employees competence descriptions need the full expressiveness of $\mathcal{ELHN}$ to convey all the embedded semantics. We therefore in the following refer to such a DL for modeling our case study.

It is easy to observe that the only characteristic shared by the four employees of our tiny case study is "an advanced knowledge about programming". Such feature might obviously be too generic to be considered for Core Competence identification. The management of a company needs instead to take into account features shared by *significant* subsets of the collection made up by the employees; the minimum required number of employees may be set by the management on the basis of a organizational needs. As an example, if the management accept that three employees have to hold some knowledge to consider it part of Core Competence, we can state that the company has advanced knowledge about object oriented programming as Core Competence. Such a result is more significant than the first one w.r.t. to the objective of determining the fields of excellence of the company.

Of course, the more the extracted knowledge is specific and unknown to the management, the more the automated

Employee Name	Concept Description Representing Employee Profile
Antonio	CSEngineer ⊓ ∃advancedKnowledge.(Java ⊓ (≥ 3 hasExperienceYears)) ⊓ ∃advancedKnowledge.Unix ⊓ ∃advancedKnowledge.(UML ⊓ (≥ 3 hasExperienceYears)) ⊓ ∃basicKnowledge.InformationSystems
Claudio	ManagerialEngineer ⊓ ∃basicKnowledge.SAP ⊓ ∃advancedKnowledge.(Java ⊓ (≥ 2 hasExperienceYears)) ⊓ ∃basicKnowledge.SoftwareEngineering
Roberto	AssetManager ⊓ ∃advancedKnowledge.C++ ⊓ ∃advancedKnowledge.(HumanResourcesManagement ⊓ (≥ 5 hasExperienceYears)) ⊓ ∃advancedKnowledge.VBscript
Daniele	Engineer ⊓ ∃advancedKnowledge.(C ⊓ (≥ 5 hasExperienceYears)) ⊓ ∃basicKnowledge.DBMS ⊓ ∃advancedKnowledge.(Javascript ⊓ (≥ 3 hasExperienceYears))

Table 4: The Concept Descriptions representing Employee Profiles

process we propose is useful for achieving competitive advantage.

Observe that, even though the services proposed in this paper have been specifically devised for solving Core Competence extraction needs, they are helpful in all frameworks in which a partial sharing of features in a collection of resources is required. Imagine for example the scenario of a medical research laboratory investigating new diseases. If patient health records are formalized in DL w.r.t. an ontology describing health care domain, the symptoms of patients sharing the final diagnosis may be investigated to find common features. Also in such a scenario a partial coverage of patients collections might be a crucial information for detecting symptoms commonly associated to a disease, which could be therefore helpful in diagnosis process.

4. INFERENCE SERVICES

In the following we start recalling standard services we use in our approach and then proceed to introduce proposed ones. The most important —and well-known— service characterizing reasoning in DL checks for specificity hierarchies, by determining whether a concept description is more specific than another one or, formally, if there is a *subsumption* relation between them.

DEFINITION 1 (SUBSUMPTION). *Given two concept descriptions C and D and a TBox $\mathcal{T}$ in a DL $\mathcal{L}$, we say that D subsumes C w.r.t. $\mathcal{T}$ if for every model of $\mathcal{T}$, $C^{\mathcal{I}} \subset D^{\mathcal{I}}$. We write $C \sqsubseteq_{\mathcal{T}} D$, or simply $C \sqsubseteq D$ if we assume an empty TBox.*

For example, consider the following concept descriptions, referred to a competence and an employee profile, respectively:

- $C_1 = \exists$basicKnowledge.Programming
- $P_1 = \exists$basicKnowledge.Java $\sqcap \exists$hasMasterDegree.ComputerScience $\sqcap$ Consultant

Knowledge expressed by P_1 is more specific than the one required by T_1: according to the previous definition C_1 subsumes P_1.

Based on subsumption new reasoning services may be defined in DLs. In particular, all of the following services are aimed at finding commonalities in collections of concepts formalized in a generic DL $\mathcal{L}$.

We recall Least Common Subsumer definition by Cohen and Hirsh [10], before introducing new services based on it.

DEFINITION 2 (LCS, [10]). *Let $C_1, \ldots, C_n$ be n concept descriptions in a DL $\mathcal{L}$. An LCS of $C_1, \ldots, C_n$, denoted by $LCS(C_1, \ldots, C_n)$, is a concept description E in $\mathcal{L}$ such that the following conditions hold:*

(i) *$C_i \sqsubseteq E$ for $i = 1, \ldots, n$*

(ii) *E is the least $\mathcal{L}$-concept description satisfying (i),* i.e., *if E' is an $\mathcal{L}$-concept description satisfying $C_i \sqsubseteq E'$ for all $i = 1, \ldots, n$, then $E \sqsubseteq E'$*

It is well known that, if the DL $\mathcal{L}$ admits conjunction of concepts "$\sqcap$", then the LCS is unique up to concept equivalence (since if both E_1 and E_2 are common subsumers of $C_1, \ldots, C_n$, then so is $E_1 \sqcap E_2$). Moreover, if union of concepts "$\sqcup$" is allowed in $\mathcal{L}$, then for every set of concepts $C_1, \ldots, C_n \in \mathcal{L}$, their LCS is $C_1 \sqcup \cdots \sqcup C_n$. Hence, the study of LCS is limited to DLs not admitting union.

In order to deal with partial commonalities, we define now common subsumers of k concepts in a collection of n elements.

DEFINITION 3 (K-CS). *Let $C_1, \ldots, C_n$ be n concepts in a DL $\mathcal{L}$, and let be $k < n$. A* k-Common Subsumer *(k-CS) of $C_1, \ldots, C_n$ is a concept $D \not\equiv \top$ such that D is an LCS of k concepts among $C_1, \ldots, C_n$.*

By definition, LCSs are also k-CSs, for every $k < n$. For this reason we define a particular subset of k-CSs, adding informative content to the LCS computation.

DEFINITION 4 (IKCS). *Let $C_1, \ldots, C_n$ be n concepts in a DL $\mathcal{L}$, and let $k < n$. An* Informative k-Common Subsumer *(IkCS) of $C_1, \ldots, C_n$ is a k-CS E such that E is strictly subsumed by $LCS(C_1, \ldots, C_n)$.*

In the following we define concepts subsuming the maximum number of elements in a collection:

DEFINITION 5 (BCS). *Let $C_1, \ldots, C_n$ be n concepts in a DL $\mathcal{L}$. A* Best Common Subsumer *(BCS) of $C_1, \ldots, C_n$ is a concept S such that S is a k-CS for $C_1, \ldots, C_n$, and for every $k < j \leq n$ every j-CS $\equiv \top$.*

The Least Common Subsumer, when not equivalent to the universal concept, is of course the best common subsumer a collection may have: it subsumes the whole collection. As a consequence, the computation of BCSs for collections admitting LCSs not equivalent to $\top$ is meaningless. For such collections, we alternatively propose the following service:

DEFINITION 6 (BICS). *Let $C_1, \ldots, C_n$ be n concepts in a DL $\mathcal{L}$. A* Best Informative Common Subsumer *(BICS) of $C_1, \ldots, C_n$ is a concept B such that B is an Informative k-CS for $C_1, \ldots, C_n$, and for every $k < j \leq n$ every j-CS is not informative.*

PROPOSITION 1. *If $LCS(C_1, \ldots, C_n) \equiv \top$, every BCS is also a BICS.*

Even though the services defined above may appear quite similar to each other at a first sight, it has to be underlined that they deal with different problems [12]:

- k-CS: can be computed for every collection of elements and finds least common subsumers of k elements among the n belonging to the collection;
- IkCS: describes those k-CSs adding an informative content to the one provided by LCS, *i.e.*, more specific than LCS. Observe that IkCS does not exist when every subset of k concepts has the same LCS as the one of all $C_1, \ldots, C_n$;
- BICS: describes IkCSs subsuming h concepts, such that h is the maximum cardinality of subsets of the collection for which an IkCS exists. A BICS does not exist if and only if $C_i \equiv C_j$ for all $i, j = 1, \ldots, n$;
- BCS: may be computed only for collections admitting only LCS equivalent to the universal concept; it finds k-CSs such that k is the maximum cardinality of subsets of the collection for which an LCS not equivalent to $\top$ exists.

5. COMMONALITIES EXTRACTION

In the following we show how to find commonalities in concept collections formalized in DL in accordance with outlined services, while determining some complexity results for such services.

In the computation of common subsumers of a collection of concept descriptions $C_1, \ldots, C_n$ we assume that all concepts C_i in the collection are consistent; hence $C_i \not\equiv \bot$ for every $C_i \in (C_1, \ldots, C_n)$.

The reasoning services introduced in Section 4 ask for the concepts of the input collection to be written in components according to the following recursive definition:

DEFINITION 7 (CONCEPT COMPONENTS). *Let C be a concept description in a DL $\mathcal{L}$, with C written in a conjunction $C^1 \sqcap \cdots \sqcap C^m$. The* Concept Components *of C are defined as follows:*

1. *if C^j, with $j = 1 \ldots, m$ is either a concept name, or a negated concept name, or a number restriction, then C^j is a* Concept Component *of C*
2. *if $C^j = \exists R.D$, with $j = 1 \ldots, m$, then $\exists R.\top$ is a* Concept Component *of C*
3. *if $C^j = \forall R.E$, with $j = 1 \ldots, m$, then $\forall R.E^k$ is a* Concept Component *of C, for each E^k* Concept Component *of E*

Observe that we do not propagate universal restriction over existential restriction since existential restriction always simplify to a component of the form $\exists R.\top$.

For the computation of the sets of k-CSs, IkCSs, BICSs and BCSs of a collection of concepts we define in the following a *Subsumers Matrix*, for the representation of the collection itself.

DEFINITION 8 (SUBSUMERS MATRIX). *Let $C_1, \ldots, C_n$ be a collection of concept descriptions C_i in a Description Logic $\mathcal{L}$ and let $D_j \in \{D_1, \ldots, D_m\}$ be the* Concept Components *deriving from all concepts in the collection. We define the* ***Subsumers Matrix*** *$S = (s_{ij})$, with $i = 1 \ldots n$ and $j = 1 \ldots m$, such that $s_{ij} = 1$ if the component D_j subsumes C_i, and $s_{ij} = 0$ if the component D_j does not subsume C_i.*

DEFINITION 9. *Referring to the Subsumers Matrix of $C_1, \ldots, C_n$, we define:*

Concept Component Signature (sig_{D_j}) *: set of indices of concepts $C_1, \ldots, C_n$ subsumed by D_j; observe that $sig_{D_j} \subseteq \{1, \ldots, n\}$;*

Concept Component Cardinality (T_{D_j}) *: cardinality of sig_{D_j}, that is, how many concepts among $C_1, \ldots, C_n$ are subsumed by D_j. Such a number is $\sum_{i=1}^{n} s_{ij}$;*

Maximum Concept Component Cardinality *(M_S): maximum among all concept component cardinalities, that is, $M_S = max\{T_{D_1}, \ldots T_{D_m}\}$;*

Second Maximum Concept Component Cardinality *(PM_S): maximum among the cardinalities of concept components not subsuming all n concepts in the collection ($PM_S = max\{T_{D_j} | T_{D_j} < n\}$); by definition $PM_S < n$;*

Common Signature Class($\bigcap_{sig_{D_j}}$) *: concept formed by the conjunction of all concept components whose signature contains D_j: $\sqcap\{D_h \mid sig_{D_j} \subseteq sig_{D_h}\}$*

Definition 8 hints that the computation of Subsumers Matrix includes an oracle to subsumption. As a consequence the following proposition holds:

PROPOSITION 2. *Let $\mathcal{L}$ be a DL whose subsumption problem is decidable in polynomial time. Then Subsumers Matrix in $\mathcal{L}$ is computable in polynomial time too.*

Such a result causes the computation of common subsumers in DLs with different complexities for subsumption to be treated separately. We therefore concentrate on the DL needed for modeling our case study, $\mathcal{ELHN}$, in Section 5.1 and provide results for different DLs in Section 5.2.

Nevertheless some considerations are logic independent and preliminary to the determination of common subsumers in every DL.

Firstly, we define the solution sets for the introduced reasoning services, regardless of the DL employed for the representation of concepts in a given collection:

	D_1	D_2	$D\ldots$	D_m
C_1	x	x		x
C_2		x		x
$C\ldots$				x
C_n	x	x		x
	IkCS, k-CS	*k-CS, IkCS, BICS*		*LCS, k-CS*

Figure 3: Subsumers Matrix of a collection admitting LCS $\neq \top$

	D_1	D_2	$D\ldots$	D_m
C_1	x	x		x
C_2		x		x
$C\ldots$				
C_n	x			x
	k-CS, IkCS	*k-CS, IkCS*		*k-CS, IkCS, BICS, BCS*

Figure 4: Subsumers Matrix of a collection whose LCS $\equiv \top$

$\underline{BCS}$: set of BCSs of the collection

$\underline{BICS}$: set of BICSs of the collection

$\underline{ICS}_k$: set of IkCSs of the collection, given $k < n$

$\underline{CS}_k$: set of k-CSs of the collection, given $k < n$

PROPOSITION 3. *Given a DL $\mathcal{L}$ and a collection of concept descriptions in $\mathcal{L}$, for each $k < n$ the solution sets of the collection are such that $I_k \subseteq L_k$. If the collection admits only the universal concept as LCS, then $B = BI$ also holds.*

PROPOSITION 4. *The following observations on the Subsumers matrix represent necessary conditions to determine common subsumers of the collection:*

1. *Only concept components D_j for which $T_{D_j} \geq k$ are meaningful for the determination of $\underline{CS}_k$ elements*
2. *Only concept components subsuming k concepts but not all n concepts in the collection are meaningful for the determination of $\underline{ICS}_k$ elements: D_j for which $k \leq T_{D_j} < n$*
3. *Only concept components D_j for which $T_{D_j} = PM_S$ are taken into account for the determination of $\underline{BICS}$ elements*
4. *Only concept components D_j for which $T_{D_j} = M_S$, with $M_S < n$ may determine $\underline{BCS}$ elements.*

The representation (for $k = 2$) in Figure 3 and 4 shows the introduced necessary conditions and should clarify the differences among introduced services. In particular, notice that in case of a collection admitting LCS $\neq \top$ (Figure 3), the best common subsumer of the collection is the LCS itself, so only the computation of BICSs makes sense. On the contrary, for a collection admitting only LCS $\equiv \top$ (Figure 4), BCSs may be computed and are BICSs too.

5.1 Computation in $\mathcal{ELHN}$

The commonalities extraction process in the DL we employ to model our case study, relies on computation results for LCS computation. Baader et al. [4] showed that, even for the small DL $\mathcal{EL}$, the shortest representation of the LCS of n concepts has exponential size in the worst case, and this result holds also when a TBox is used to shorten possible repetitions [7]. Such a result affects the computation of the introduced solution sets of common subsumers, as stated in the following theorem.

THEOREM 1. *The computation of the solution sets $\underline{BCS}$, $\underline{BICS}$, $\underline{CS}_k$, $\underline{ICS}_k$ for a collection of concept descriptions in $\mathcal{ELHN}$ may be reduced to the problem of computing the LCS of the subsets of the collection.*

PROOF. For computing $\underline{CS}_k$ it is sufficient to compute for every subset $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ the concept $LCS(C_{i_1}, \ldots, C_{i_k})$.
The same holds for $\underline{ICS}_k$, excluding those $LCS(C_1, \ldots, C_k)$ which are equivalent to $LCS(C_1, \ldots, C_n)$. For the computation of the sets $\underline{BCS}$ and $\underline{BICS}$, instead, we provide an algorithm that uses the one proposed by Kusters and Molitor [17] for LCS computation. The algorithm takes as input the collection $C_1, \ldots, C_n$ represented through its Subsumers Matrix. Consider now the Concept Components of the elements C_i in the collection: the reduction in Step 2 of Definition 7 causes not all the components to be straightly included in the solution sets $\underline{BCS}$ and $\underline{BICS}$. For example, consider the concept description
$C_1 =$ `AssetManager` $\sqcap \exists$`basicKnowledge.Psychology`: the resulting concept component is $D_1 = \exists$`basicKnowledge.`$\top$ and $D_2 = \exists$`advancedKnowledge.`$\top$ (taking also into account the TBox definitions in Figure 2). Even though such component is selected for the determination of the solution sets according to Proposition 4, it just individuates the concepts in the collection to consider for the determination of $\underline{BCS}$ and $\underline{BICS}$. For each component D_j we denote LCS_{D_j} the LCS of the C_i such that $s_{ij} = 1$.

Input : Subsumers Matrix $S = (s_{ij})$ for a collection of concepts $C_i \in \{C_1, \ldots, C_n\}$ in $\mathcal{ELHN}$
Output: $\underline{BICS}$; $\underline{BCS}$
1 **if** $M_S = n$ **then**
2 $\underline{BCS} := \emptyset$;
3 **foreach** D_j *s.t.* $T_{D_j} = PM_S$ **do** $\underline{BICS} := \underline{BICS} \cup LCS_{D_j}$;
4 **else**
5 **foreach** D_j *s.t.* $T_{D_j} = M_S$ **do**
6 $\underline{BCS} := \underline{BCS} \cup LCS_{D_j}$;
7 $\underline{BICS} := \underline{BCS}$;
8 **return** $\underline{BCS}$, $\underline{BICS}$;
Algorithm 1: An algorithm for Common Subsumers enumeration in $\mathcal{ELHN}$

Algorithm 1 requires the computation of the LCS of l concepts — with $l \leq n$— in lines 3, 6. Similarly to the approach used in [5] we limit the size on the input collection from n to l, and we compute the LCS of the l concepts as shown in [17]. The problem of determining the solution sets

of a collection may be then reduced to the computation of the LCS of subsets of the collection itself. □

5.2 Further Computation Results

The choice of proposing an algorithm finding informative commonalities in $\mathcal{ELHN}$ is due to modeling needs in our case study. We therefore generalize our approach by investigating the computational complexity of introduced services in two different DLs, namely $\mathcal{ALN}$ and $\mathcal{ALE}$.

The computational complexity for $\mathcal{ALN}$ is lower than the one for $\mathcal{ELHN}$, given that $\mathcal{ALN}$ lacks of existential restrictions, which is the source of exponentiality [7]. In particular, in knowledge domains which can be modeled in $\mathcal{ALN}$ the introduced common subsumers can be computed by straightforwardly applying conditions in Proposition 4, as stated in the following theorem.

THEOREM 2. *For a collection of concept descriptions in $\mathcal{ALN}$ the necessary conditions for solution sets determination exposed in Proposition 4 are also sufficient.*

PROOF. Let the Common Signature Class $\bigcap_{sig_{D_j}}$ of a concept component D_j satisfy one of the necessary conditions in Proposition 4 and let LCS_{D_j} be the LCS of the k concepts subsumed by D_j. If $\bigcap_{sig_{D_j}}$ is just a common subsumer and not the least one, $LCS_{D_j} \sqsubset \bigcap_{sig_{D_j}}$, hence LCS_{D_j} must be equivalent to $\bigcap_{sig_{D_j}} \sqcap F$, with F a suitable concept. This means that F subsumes each concept C_i subsumed by D_j hence, according to Definition 7, F must be one of the concept components conjoined in $\bigcap_{sig_{D_j}}$ for the computation of $\underline{BCS}$, $\underline{BICS}$, $\underline{CS}_k$ and $\underline{ICS}_k$ elements. □

When a TBox is present—even a simple one, made of axioms $A \doteq C$ where A is a name—Nebel proved that subsumption is PSPACE-hard even for the simple DL $\mathcal{FL}_0$ [21][1]. Hence, in the following complexity analysis we decouple the contribution of the subsumption tests in the subsumers matrix computation from the computation of different introduced common subsumers.

The complexity of solution sets computation is object of the following theorem.

THEOREM 3. *Let $C_1, \ldots, C_n$, $\mathcal{T}$ be n concepts and a simple Tbox in $\mathcal{ALN}$, let m be the sum of the sizes of $C_1, \ldots, C_n$, and let $S(s)$ be a monotone function bounding the cost of deciding $C \sqsubseteq_{\mathcal{T}} D$ in $\mathcal{ALN}$, whose argument s is $|C|+|D|+|\mathcal{T}|$. The computation of the solution sets $\underline{BCS}$, $\underline{BICS}$, $\underline{CS}_k$, $\underline{ICS}_k$ for a collection of concept descriptions in $\mathcal{ALN}$ is then a problem in $O(m^2 + (S(m))^2)$.*

PROOF. We propose an algorithm determining the sets $\underline{BICS}$, $\underline{CS}_k$, $\underline{ICS}_k$, $\underline{BCS}$ of a collection $\{C_1, \ldots, C_n\}$ of concepts in $\mathcal{ALN}$, whose Subsumers Matrix is given as input. Hence the computation of the Subsumers Matrix can be carried over in polynomial time (see Proposition 2).

[1]However, Nebel himself claims that exponentiality raises from the nesting of the definitions (a concept that defines another that defines another etc.) and that for "bushy but not deep" TBoxes exponentiality does not arise. A precise characterization of what "bushy but not deep" means has been given by Di Noia et al. [14].

According to Theorem 2 the determination of the output sets asks then only for the enumeration of Common Signature Classes of the components D_j chosen according to conditions in Proposition 4.

The algorithm for Common Subsumers enumeration in $\mathcal{ALN}$ is shown in the following:

Input : Collection Subsumers Matrix $S = (s_{ij})$ for a collection of concepts $\{C_1, \ldots, C_n\}$ in $\mathcal{ALN}$, integer $k < n$
Output: $\underline{CS}_k$; $\underline{ICS}_k$; $\underline{BICS}$; $\underline{BCS}$

```
1  CS_k := ∅; ICS_k := ∅; BICS := ∅; BCS := ∅;
2  foreach D_j s.t. T_{D_j} ≥ k do
3      CS_k := CS_k ∪ ⋂_{sig_{D_j}};
4      if T_{D_j} < n then ICS_k := ICS_k ∪ ⋂_{sig_{D_j}};
5  if M_S = n then
6      foreach D_j s.t. T_{D_j} = PM_S do
7          BICS := BICS ∪ ⋂_{sig_{D_j}};
8  else foreach D_j s.t. T_{D_j} = M_S do
9      BCS := BCS ∪ ⋂_{sig_{D_j}}; BICS := BICS ∪ ⋂_{sig_{D_j}};
10 return CS_k, BCS , ICS_k, BICS;
```

Algorithm 2: An algorithm for Common Subsumers enumeration in $\mathcal{ALN}$

It is straightforward to verify that the algorithm runs in $O(m^2)$. Given that subsumers matrix can be computed in $O(m^2 + (S(m))^2)$, the claim follows. □

For the computation of $\underline{BICS}$ and $\underline{BCS}$ in $\mathcal{ALE}$, we can use Algorithm 1. For computing $\underline{CS}_k$ it is again sufficient to compute for every subset $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$ the concept $LCS(C_{i_1}, \ldots, C_{i_k})$. The same holds for $\underline{ICS}_k$, excluding those $LCS(C_1, \ldots, \mathcal{C}_k)$ which are equivalent to $LCS(C_1, \ldots, \mathcal{C}_n)$. A method for computing the hierarchy of all such LCSs (but for the DL $\mathcal{ALC}$, so without number restrictions and role inclusions) has been proposed by Baader and Sertkaya[6]. Once such a hierarchy is computed, it is easy to find IkCSs, BICSs, and BCSs. However, we observe that for DLs in Table 1, some theoretical results on LCS are still missing. Since the output of an algorithm computing an LCS in DLs including $\mathcal{EL}$ is exponential in the worst case [7], we refer to the usual model of computation of Turing Machines with three tapes: one for input (read-only), one for working memory, and one for output (write-only) [16]. In such a model, space complexity refers to the working memory tape. In this setting we observe that the algorithm proposed by Baader and Turhan [7] for LCS in $\mathcal{ALE}$ works in polynomial space, while the algorithm proposed by Kusters and Molitor [17] works in exponential space, since it relies on a normalization step for input concepts, and it is still unclear if such a normalization step can be avoided.

6. CASE STUDY SOLUTION

In order to better clarify the proposed services we apply the commonalities extraction process detailed in Section 5 to our tiny example scenario proposed in Section 3.

The input collection therefore is made up by the four profiles in Table 4. Let 50 % be the required level of competence coverage set by company management to individuate Core Competence. We are hence interested in determining competence shared by at least two employees out of the four in the company.

	D_1	D_2	D_3
Antonio	x	x	x
Claudio	x	x	x
Roberto		x	
Daniele	x	x	x

Figure 5: Example Collection Subsumers Matrix

In order to compute the set $\underline{CS}_2$, it is sufficient to compute the LCS of all subsets of cardinality 2:

$LCS(Antonio, Claudio) =$
$\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Java} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.SoftwareEngineering} \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}$

$LCS(Antonio, Roberto) = \exists \texttt{advancedKnowledge.OOP}$

$LCS(Antonio, Daniele) =$
$\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 5\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems})$

$LCS(Claudio, Roberto) = \exists \texttt{advancedKnowledge.OOP}$

$LCS(Claudio, Daniele) =$
$\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}$

$LCS(Roberto, Daniele) =$
$\exists \texttt{advancedKnowledge.ScriptLanguages}$

All elements in $\underline{CS}_2$ have to be investigated w.r.t. the LCS of the collection to identify Informative 2-Common Subsumers. We need then to compute such a concept:

$LCS = LCS(Antonio, Claudio, Roberto, Daniele) =$
$\exists \texttt{advancedKnowledge.Programming}$

Each element in $\underline{CS}_2$ is more specific than LCS and then belongs also to $\underline{ICS}_2$.

We use instead Algorithm 1 for computing the sets $\underline{\text{BCS}}$ and $\underline{\text{BICS}}$. The algorithm requires the concept collection Subsumers Matrix as input; so we have to compute it first. The concept components coming from the collection are computed according to Definition 7 and take into account TBox definitions in Figure 2. As a result we have the three concept components as in the following:

- $D_1 = \exists \texttt{hasMasterDegree.}\top$
- $D_2 = \exists \texttt{advancedKnowledge.}\top$
- $D_3 = \exists \texttt{basicKnowledge.}\top$

The collection subsumers matrix is shown in Figure 5 and is characterized by the following values: $M_S = 4$, $PM_S = 3$. By applying Algorithm 1 we have that $\underline{BCS} := \emptyset$ (line 2) and we need to compute, according to line 3, LCS_{D_1} and LCS_{D_3}. It is straightforward to notice that $LCS_{D_1} \equiv LCS_{D_3}$, so the only BICS for the collection is:

$LCS(Antonio, Claudio, Daniele) =$
$\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}$

The solution sets coming from the commonalities extraction process are detailed below:

$\underline{BCS} = \emptyset$

$\underline{BICS} =$

$\{\ (\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems})\ \}$

$\underline{CS}_2 =$

$\{\ (\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Java} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.SoftwareEngineering} \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}),$

$(\exists \texttt{advancedKnowledge.OOP}),$

$(\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 5\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}),$

$(\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge.}(\texttt{Programming} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge.InformationSystems}),$

$(\exists \texttt{advancedKnowledge.ScriptLanguages})\ \}$

$\underline{ICS}_2 = \underline{CS}_2$

Thanks to the commonalities extraction process proposed here, the company in our tiny case study discovered some new information about the fields of excellence characterizing its know-how. In particular, by computing the LCS of the collection of employee profiles, the company management may discover that the whole personnel knows Programming at an advanced level, which is quite a generic sort of information, probably well known by the company.

More significant and unknown commonalities may be found by computing the set of IkCSs: *i)* knowledge embedded in Engineer job title together with an advanced knowledge in Java, related to two years of working experience, and a basic knowledge in Information Systems and software engineering; *ii)* advanced knowledge about object oriented programming;*iii)* advanced knowledge about script languages. Such commonalities are therefore informative w.r.t. the objective of determining unknown fields of excellence in the company.

Moreover the company may discover, thanks to the computation of BICSs, that knowledge shared by the maximum number of employees in the company (excluding Programming which is shared by all employees) is an advanced knowledge of programming related to 2 years of experience, together with the knowledge embedded in Engineer job title and a basic knowledge in Information Systems.

In large knowledge intensive companies, like multinational ones, the proposed approach may hence help to detect *hidden* fields of excellence of a company, especially if out of its core business, thus representing a potential source of competitive advantage.

7. CONCLUSIONS

Motivated by the need to identify and extract so called Core Competence in knowledge intensive companies, and by the limits of LCS in such a framework, we have introduced

and exploited informative common subsumers in Description Logics, useful in application fields where there is the need to extract significant informative commonalities in concept collections, and such commonalities are not shared by the entire collection. We have proposed definitions, algorithms to compute such informative common subsumers for various DLs and presented simple complexity results.

Obviously our approach requires competencies —or in general concept collections— be modeled in accordance with an ontology, but as semantic-based languages and technologies gain momentum it is reasonable to assume that more and more companies will move towards a logic-based formalization of their skills and processes and be able to take advantage of proposed and other relevant non-standard services.

Acknowledgment

We thank Franz Baader and Sergei O. Kuznetsov for helpful discussions. This work has been supported in part by EU-FP6-IST-26896 project and Apulia Region funded projects PE_013 *Innovative models for customer profiling* and PS_092 *DIPIS*.

8. REFERENCES

[1] F. Baader, D. Calvanese, D. Mc Guinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.

[2] F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*, Lecture Notes in Artificial Intelligence. Springer-Verlag, 2003.

[3] F. Baader and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proceedings of the Twenty second German Annual Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140, Bremen, Germany, 1998. Springer–Verlag.

[4] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumer in description logics with existential restriction. Technical Report LTCS-Report 98-09, RWTH Aachen, 1998.

[5] F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *Proceedings of the Seventh International Conference on Conceptual Structures (ICCS'00)*, pages 292–305, London, UK, 2000. Springer-Verlag.

[6] F. Baader and B. Sertkaya. Applying formal concept analysis to description logics. In P. Eklund, editor, *Proceedings of the 2nd International Conference on Formal Concept Analysis (ICFCA 2004)*, volume 2961 of *Lecture Notes in Artificial Intelligence*, pages 261–286. Springer, 2004.

[7] F. Baader and A.-Y. Turhan. On the problem of computing small representations of least common subsumers. In *Proceedings of the Twenty sixth German Annual Conference on Artificial Intelligence (KI'02)*, volume 2479 of *Lecture Notes in Artificial Intelligence*, Aachen, Germany, 2002. Springer-Verlag.

[8] A. Borgida, R. Brachman, D. L. McGuinness, and L. A. Resnick. CLASSIC: A structural data model for objects. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 59–67, 1989.

[9] W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In P. Rosenbloom and P. Szolovits, editors, *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 754–761, Menlo Park, California, 1993. AAAI Press.

[10] W. Cohen and H. Hirsh. Learning the CLASSIC description logics: Theorethical and experimental results. In *Proceedings of the Fourth International Conference on the Principles of Knowledge Representation and Reasoning (KR'94)*, pages 121–133, 1994.

[11] S. Colucci, T. Di Noia, E. Di Sciascio, F. Donini, and A. Ragone. Semantic-based skill management for automated task assignment and courseware composition. *Journal of Universal Computer Science*, 13(9):1184–1212, 2007.

[12] S. Colucci, E. Di Sciascio, and F. Donini. Partial and informative common subsumers of concepts collections in description logics. In *Proc. of the 21st Intl. Workshop on Description Logics (DL'08)*, 2008.

[13] DAML+OIL. DAML+OIL Specifications. www.daml.org/2001/03/daml+oil-index.html, 2001.

[14] T. Di Noia, E. Di Sciascio, and F. Donini. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research*, 29:269–307, 2007.

[15] G. Hamel and C. K. Prahalad. The core competence of the corporation. *Harvard Business Review*, May-June:79–91, 1990.

[16] D. S. Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, pages 67–161. 1990.

[17] R. Küsters and R. Molitor. Structural Subsumption and Least Common Subsumers in a Description Logic with Existential and Number Restrictions. *Studia Logica*, 81:227–259, 2005.

[18] C. C. Markides and P. J. Williamson. Related diversification, core competences and corporate performance. *Strategic Management Journal*, 15:49–65, 1994.

[19] D. McGuinness, R. Fikes, J. Hendler, and L. Stein. DAML+OIL: An Ontology Language for the Semantic Web . *IEEE Intelligent Systems*, 17(5):72–80, 2002.

[20] R. Möller, V. Haarslev, and B. Neumann. Semantics-based information retrieval. In *Proceedings of the International Conference on Information Technology and Knowledge Systems (IT&KNOWS'98)*, Vienna, Budapest, 1998.

[21] B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1990.

[22] R. R. Nelson. Why do firms differ, and how does it matter? *Strategic Management Journal*, 12:61–74, 1991.

[23] OWL. www.w3.org/TR/owl-features/, 2004.