Semantic-based Automated Evaluation of Company Core Competence

Simona Colucci[1 3], Eugenio Di Sciascio[1], Francesco M. Donini[2], and Eufemia Tinelli[3 4]

[1] SisInfLab–Politecnico di Bari, Bari, Italy
[2] Università della Tuscia, Viterbo, Italy
[3] D.O.O.M. s.r.l., Matera, Italy
[4] Università degli Studi di Bari, Bari, Italy

Abstract. Core Competence evaluation is crucial for strategical choices in knowledge intensive companies. Such a process is usually manually performed by the management on the basis of subjective criteria, which can then cause non-optimal decisions, especially in wide companies. We propose here a semantic-based approach for the automatic evaluation of Core Competence, exploiting novel reasoning services in Description Logics, extracting commonalities in a collection of resource descriptions. Such inferences aim at identifying features shared at least by a significant portion of a collection of professional profiles formalized in accordance with a logic language. We are in fact not necessarily interested in competence shared by the whole company personnel.

1 Introduction

In [11] 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 [13, 15].

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. If employee profiles and organization knowledge are formalized in a collection of concept descriptions according to a formal language conveying semantics, the management could extract company Core Competence by searching for features shared in such a collection. In particular, Description Logics [1] offer inference services specifically aimed at identifying concept collection commonalities. Least Common Subsumers(LCS) have in fact been defined [7] — originally for the DL underlying Classic [6]— 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, such as the bottom-up construction of knowledge bases [2], inductive learning algorithms [8] and information retrieval [14].

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 Core Competence evaluation, 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. 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 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.

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 k-Common Subsumers and three more specific reasoning services needed for the commonalities extraction process, provided in Section 5. Introduced services are illustrated in the motivating organizational context 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.

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.

The motivating scenario we here propose needs at least the expressiveness of $\mathcal{ELHN}$ sublanguage of DLs for resources representation. $\mathcal{ELHN}$ allows of course for the constructors provided in every DL: *conjunction* of concepts(e.g.$\texttt{C++} \sqcap \texttt{Java}$), *negation* of atomic concepts ($\neg\texttt{Skill}$), *top* concept($\top$, interpreted as the whole domain) and *bottom* concept ($\bot$, interpreted as the empty set). The following constructors are furthermore available: existential restriction (e.g. $\exists\texttt{knowsLanguage}.\texttt{English}$), num-

ber restrictions (e.g. $(\leq\ 2\ \texttt{knowsLanguage})$ and $(\geq\ 3\ \texttt{hasExperienceYears})$) and role inclusion (e.g. $\texttt{advancedKnowledge} \sqsubseteq \texttt{basicKnowledge}$).

3 Motivating Case Study

The investigations on the problem of evaluating Core Competence in knowledge intensive companies 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. by D.O.O.M. s.r.l.

IMPAKT is specifically aimed at semantic-based human resources management [9] and provides Core Competence extraction as decisional support service.

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.

$$
\begin{aligned}
&\texttt{Operations Management} \sqsupseteq \left\{ \begin{array}{l} \texttt{OperationsOptimization} \\ \texttt{Process Management} \\ \texttt{ProductionManagement} \end{array} \right. \\
&\texttt{SoftwareEngineering} \sqsupseteq \{ \texttt{UML} \\
&\texttt{Programming} \sqsupseteq \left\{ \begin{array}{l} \texttt{ScriptLanguages} \sqsupseteq \left\{ \begin{array}{l} \texttt{Javascript} \\ \texttt{VBscript} \end{array} \right. \\ \texttt{OOP} \sqsupseteq \left\{ \begin{array}{l} \texttt{Java} \\ \texttt{C++} \end{array} \right. \\ \texttt{StrucProgramming} \sqsupseteq \texttt{C} \end{array} \right. \\
&\texttt{InformationSystems} \sqsupseteq \left\{ \begin{array}{l} \texttt{DBMS} \\ \texttt{ERPsystem} \sqsupseteq \{ \texttt{SAP} \\ \texttt{TCP/IP} \end{array} \right. \\
&\texttt{OperatingSystems} \sqsupseteq \left\{ \begin{array}{l} \texttt{Unix} \\ \texttt{Windows} \end{array} \right. \\
&\texttt{AssetAllocation} \sqsupseteq \texttt{HumanResourcesManagement} \\
&\texttt{basicKnowledge} \sqsupseteq \texttt{advancedKnowledge}
\end{aligned}
$$

Fig. 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:

$\texttt{Manager} \equiv \exists \texttt{advancedKnowledge}.(\texttt{ManagementTechniques} \sqcap (\geq\ 8\ \texttt{hasExperienceYears}))$
$\texttt{AssetManager} \equiv$
$\quad \texttt{Manager} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{AssetAllocation} \sqcap (\geq\ 5\ \texttt{hasExperienceYears}))$
$\texttt{Engineer} \equiv \exists \texttt{advancedKnowledge}.(\texttt{Design} \sqcap (\geq\ 5\ \texttt{hasExperienceYears})) \sqcap$
$\quad \exists \texttt{hasMasterDegree}.\texttt{Engineering} \sqcap \exists \texttt{basicKnowledge}.\texttt{OperationsOptimization}$
$\texttt{ManagerialEngineer} \equiv \texttt{Engineer} \sqcap \exists \texttt{basicKnowledge}.\texttt{ERPsystem} \sqcap$
$\quad \exists \texttt{advancedKnowledge}.(\texttt{ProductionManagement} \sqcap (\geq\ 3\ \texttt{hasExperienceYears})) \sqcap$
$\quad \exists \texttt{advancedKnowledge}.(\texttt{Process Management} \sqcap (\geq\ 3\ \texttt{hasExperienceYears}))$
$\texttt{CSEngineer} \equiv \texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge}.\texttt{OperatingSystems} \sqcap$
$\quad \exists \texttt{advancedKnowledge}.(\texttt{Programming} \sqcap (\geq\ 5\ \texttt{hasExperienceYears})) \sqcap$
$\quad \exists \texttt{advancedKnowledge}.(\texttt{SoftwareEngineering} \sqcap (\geq\ 3\ \texttt{hasExperienceYears}))$

Fig. 2. TBox Definitions

- **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.

The four profiles are formalized according to the given TBox as shown in the following:

Antonio = $\texttt{CSEngineer} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{Java} \sqcap (\geq\ 3\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{advancedKnowledge}.\texttt{Unix} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{UML} \sqcap (\geq\ 3\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge}.\texttt{InformationSystems}$

Claudio = $\texttt{ManagerialEngineer} \sqcap \exists \texttt{basicKnowledge}.\texttt{SAP} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{Java} \sqcap$
$(\geq\ 2\ \texttt{hasExperienceYears})) \sqcap \exists \texttt{basicKnowledge}.\texttt{SoftwareEngineering}$

Roberto =
$\texttt{AssetManager} \sqcap \exists \texttt{advancedKnowledge}.\texttt{C++} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{HumanResourcesManagement} \sqcap$
$(\geq\ 5\ \texttt{hasExperienceYears})) \sqcap \exists \texttt{advancedKnowledge}.\texttt{VBscript}$

Daniele = $\texttt{Engineer} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{C} \sqcap (\geq\ 5\ \texttt{hasExperienceYears})) \sqcap$
$\exists \texttt{basicKnowledge}.\texttt{DBMS} \sqcap \exists \texttt{advancedKnowledge}.(\texttt{Javascript} \sqcap (\geq\ 3\ \texttt{hasExperienceYears}))$

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 a 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 decisional process. As an example, if the management accepts 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 process we propose is useful for achieving competitive advantage.

4 Inference Services

In the following we start recalling standard services we use in our approach and then proceed to introduce non-standard 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.

We recall Least Common Subsumer definition by Cohen and Hirsh [8], before introducing new services based on it.

Definition 2 (LCS) [8] 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 defined [10] 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 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 defined [10] 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 $LCS(C_1, \ldots, C_n)$.

We also defined [10] 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 proposed [10] 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.

5 Commonalities Extraction

In the following we show how to find commonalities in concept collections formalized in DL in accordance with outlined 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: 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; if $C^j = \exists R.D$, with $j = 1 \ldots, m$, then $\exists R.\top$ is a *Concept Component* of C; 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 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 $PM_S < n$;

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, $\mathcal{ELHN}$, even though 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: $\underline{BCS}$, set of BCSs; $\underline{BICS}$, set of BICSs; $\underline{ICS}_k$, 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 $\underline{ICS}_k \subseteq \underline{CS}_k$. If the collection admits only the universal concept as LCS, then $B = BI$ also holds.

The commonalities extraction process in $\mathcal{ELHN}$ relies on computation results for LCS computation. Baader et al. [3] 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 [5]. 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 $LCS(C_1, \ldots, \mathcal{C}_k)$ which are equivalent to $LCS(C_1, \ldots, \mathcal{C}_n)$. For the computation of the sets $\underline{BCS}$ and $\underline{BICS}$, instead, we provide Algorithm 1 that uses the one proposed by Kusters and Molitor [12] 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 $C_1 = \texttt{AssetManager} \sqcap \exists \texttt{basicKnowledge}.\texttt{Psychology}$: the resulting concept component is $D_1 = \exists \texttt{basicKnowledge}.\top$ and
$D_2 = \exists \texttt{advancedKnowledge}.\top$ (taking also into account the TBox definitions in Figure 2). Even though such component is selected for the determination of the solution sets, 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 $s_{ij} = 1$. 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 [4] we limit the size on the input collection from n to l, and we compute the LCS of the l concepts as shown in [12]. 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.

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 $\quad \underline{BCS} := \emptyset$;
3 $\quad$ **foreach** D_j *s.t.* $T_{D_j} = PM_S$ **do** $\underline{BICS} := \underline{BICS} \cup LCS_{D_j}$;
4 **else**
5 $\quad$ **foreach** D_j *s.t.* $T_{D_j} = M_S$ **do**
6 $\quad\quad \underline{BCS} := \underline{BCS} \cup LCS_{D_j}$;
7 $\quad\quad \underline{BICS} := \underline{BCS}$;
8 **return** $\underline{BCS}$, $\underline{BICS}$;

Algorithm 1: An algorithm for Common Subsumers enumeration in $\mathcal{ELHN}$

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.

The input collection therefore is made up by the four profiles in Section 3. 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.

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

$LCS(Antonio, Claudio) =$
$\quad$ Engineer $\sqcap \exists$advancedKnowledge.(Java $\sqcap$ ($\geq$ 2 hasExperienceYears)) $\sqcap$
$\quad \exists$basicKnowledge.SoftwareEngineering $\sqcap \exists$basicKnowledge.InformationSystems
$LCS(Antonio, Roberto) = \exists$advancedKnowledge.OOP
$LCS(Antonio, Daniele) =$ Engineer $\sqcap \exists$advancedKnowledge.(Programming $\sqcap$
$\quad$ ($\geq$ 5 hasExperienceYears)) $\sqcap \exists$basicKnowledge.InformationSystems)
$LCS(Claudio, Roberto) = \exists$advancedKnowledge.OOP
$LCS(Claudio, Daniele) =$ Engineer $\sqcap \exists$advancedKnowledge.(Programming $\sqcap$
$\quad$ ($\geq$ 2 hasExperienceYears)) $\sqcap \exists$basicKnowledge.InformationSystems
$LCS(Roberto, Daniele) = \exists$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:

$LCS = LCS(Antonio, Claudio, Roberto, Daniele) = \exists$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 BCS and 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: $D_1 = \exists$hasMasterDegree.$\top$, $D_2 = \exists$advancedKnowledge.$\top$, $D_3 = \exists$basicKnowledge.$\top$.

The collection subsumers matrix is shown in Figure 3 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:

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

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

Fig. 3. Example Collection Subsumers Matrix

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 the 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 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 $\mathcal{ELHN}$ and presented simple complexity results.

Obviously our approach requires competencies 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

This work has been supported in part by and Apulia Region funded projects PE_013 *Innovative models for customer profiling*, PS_092 *DIPIS* and PS_025 *L'OrMaICT*.

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 and R. Küsters. Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In *Proc. of KI'98*, volume 1504 of *LNCS*, pages 129–140, Bremen, Germany, 1998. Springer–Verlag.
3. 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.
4. F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In *Proc. of ICCS'00*, pages 292–305, London, UK, 2000. Springer-Verlag.
5. F. Baader and A.-Y. Turhan. On the problem of computing small representations of least common subsumers. In *Proc. of KI 2002*, volume 2479 of *LNAI*, Aachen, Germany, 2002. Springer.
6. A. Borgida, R.J. Brachman, D. L. McGuinness, and L. Alperin Resnick. CLASSIC: A structural data model for objects. In *Proc. of ACM SIGMOD*, pages 59–67, 1989.
7. W. Cohen, A. Borgida, and H. Hirsh. Computing least common subsumers in description logics. In Paul Rosenbloom and Peter Szolovits, editors, *Proc. of AAAI'92*, pages 754–761, Menlo Park, California, 1992. AAAI Press.
8. W. Cohen and H. Hirsh. Learning the CLASSIC description logics: Theorethical and experimental results. In *Proc. of KR'94*, pages 121–133, 1994.
9. S. Colucci, T. Di Noia, E. Di Sciascio, F.M. 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.
10. Simona Colucci, Eugenio Di Sciascio, and Francesco M. Donini. Partial and informative common subsumers of concepts collections in description logics. In *Proceedings of the 21st International Workshop on Description Logics (DL 2008)*, volume 353. CEUR, 2008.
11. G. Hamel and C. K. Prahalad. The core competence of the corporation. *Harvard Business Review*, May-June:79–91, 1990.
12. 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.
13. C. Markides and P. J. Williamson. Related diversification, core competences and corporate performance. *Strategic Management Journal*, 15:49–65, 1994.
14. R. Möller, V. Haarslev, and B. Neumann. Semantics-based information retrieval. In *Proc. of IT & KNOWS'98*, Vienna, Budapest, 1998.
15. R. Nelson. Why do firms differ, and how does it matter? *Strategic Management Journal*, 12:61–74, 1991.