Using Formal Concept Analysis for Ontology Maintenance in Human Resource Recruitment

Dominic Looser[1] **Hui Ma**[2] **Klaus-Dieter Schewe**[1]

[1] Software Competence Center Hagenberg, Austria
dominic.looser@scch.at|kdschewe@acm.org

[2] Victoria University of Wellington, School of Engineering and Computer Science, Wellington, New Zealand,
Hui.Ma@ecs.vuw.ac.nz

Abstract

Ontologies have been proven useful for many applications by enabling semantic search and reasoning. Human resource management has recently attracted interest by researchers and practitioners seeking to exploit ontologies for improving the efficiency and effectiveness of the job recruitment process. However, the quality of semantic search and decision making intimately depends on the quality of the ontology used. Most current efforts concentrate on the development of general ontologies that find wide approval by the HR community worldwide. In order to be useful for automatic matchmaking between job offers and job seekers, such high-level ontologies need to be adequately enriched with detailed domain-specific knowledge and adapted to the particular needs of individual job markets. We present an approach for enriching and adapting an existing ontology using formal concept analysis.

1 Introduction

Ontologies play an import role for semantic search and reasoning in many different important domains, including medical science, geographic information systems, and human resource recruitment applications. Ontology is used as knowledge base that represents domain knowledge so that some work can be processed without or with little involvement of human domain experts. Human resource recruitment process is getting more and more time consuming due to the increasing number of applicants and jobs advertised on the Internet. To improve the process of human recruitment, researchers propose to use ontology-based semantic match approaches. For example, (Colucci, Noia, Sciascio, Donini, Mongiello & Mottola 2003) proposes a formal approach to ontology-based semantic matching of skills descriptions, which is based on description logic formalization and reasoning. This approach can cope with cases where there is no exact match exist and provide matching results with a certain degree of matching. (Sánchez, Martínez-Béjar, Contreras, Fernández-Breis & Nieves 2006) presents an ontology-guided search engine to provide intelligent matches between job offers and CVs. Similarity matching has been used in web-based job applications for competence matching, e.g. (Bizer, Heese, Mochol, Oldakowski, Tolksdorf, Berlin & Eckstein 2005, Colucci et al. 2003, Fazel-Zarandi, Devlin, Huang & Contractor 2011, Herder 2009). Semantic matching for job recruitment relies on ontologies that capture domain knowledge of human resource recruitment experts. Therefore, it is crucial to construct a quality ontology as a knowledge base of the system.

Building ontologies is very time consuming and involves many domain experts. The ontology hierarchy is the backbone of an ontology. In human resource domain some competence taxonomies, e.g. DISCO, O*Net, have been constructed so that it can be shared by many organisations. There is an increasing number of applications using the openly available ontologies to save the effort of building ontology from scratch. However, often the openly available standard ontologies need to be extended to include some specific domain knowledge of a particular organisation. Also, ontologies need to be maintained to reflect the change of the part the world it describes, in particular, when ontology is constructed and learned from data sets in various forms, e.g. documents or databases. Any changes to the data sets might require updates of the existing ontologies. For example, when human resource experts produce different matching results than an ontology-supported system one needs to analyse the differences and capture the new domain knowledge and use it to revise the existing ontology so that the matching results can be improved by using the revised ontology. If we manually revise the existing ontology, it is not only inefficient but also may cause inconsistencies. Therefore, it is desirable to develop an ontology maintenance approach so that an ontology can be improved in a consistent and systematic way. The goal of our research is to provide a method for revising existing ontologies for job recruit systems so that quality of ontologies can be ensured and semantic matching can be improved.

Description logics are promising tools for managing ontologies because they provide not only formal representation models but also reasoning tools. For example, Chimaera (Lammari & Métais 2004) is based on description logics. (Lammari & Métais 2004) presents some techniques for building an Is_A ontology based on the description of ontology. This approach takes as input a set of instances and a set of constraints to generate concepts and the Is_A hierarchy. In the mean time there are some works using Formal Concept Analysis (FCA) to build ontologies. FCA was first introduced in (Wille 1982). In (Prediger & Stumme 1999), FCA is used to set up a so called Conceptual Information System when database is only partially given in the beginning. It shows how description logic and formal concept analysis can be connected and enriched. In particular, it shows how to use a subsumption algorithm of DL as an 'expert' for Attribute Exploration. However, it is not clear how DL is used in the process of using FCA. (Cimiano, Hotho & Staab 2005) presents a method of automatic acquisition of taxonomies or concept hierarchies from a text corpus based on FCA.

FCA is used to produce a lattice that is then converted into a special kind of partial order that constituting a concept hierarchy. In (Jia, Newman & Tianfield 2007) FCA is applied to generate information context from a tentative domain specific scientific corpus which is then mapped to a formal ontology. A semi-automatic method to ontology design is presented in (Haav 2004), with a set of rules mapping a FCA lattice to a rule language is presented. One of the advantages of using FCA for learning taxonomies is that it produces concept lattices which allows a concept to have more than one super concept, which reflects the real life concept relations. To combine the advantages of DL and FCA, in this paper we will propose a FCA-based approach of maintaining ontology, which makes use of DL as modelling tool to model ontology of the knowledge base and use FCA to design and revise ontology.

The aim of our approach is to provide a framework for automatically updating and revising existing ontology hierarchies so that it can be used to provide better matching results in the process of human resource recruitment. In this paper we use the IT domain as an example to demonstrate our proposed approach. We will show that the revised ontology can indeed improve the quality of matching results. For that we use some existing approaches of semantic matching.

Organisation. The paper is organised as the following. In Section 2, we will present a motivating example. In Section 3 we will briefly review description logic (DL) and formal concept analysis (FCA), followed by a brief comparison between DL and FCA. In Section 4 we will present a framework that uses DL and FCA in the process of ontology maintenance. Section 5 will relate our work with existing approaches of using FCA and DL in the process of ontology design. Finally, Section 6 concludes the paper with a brief summary of our contribution.

2 Motivating Scenario

Job recruitment often involves processing a big number of applications for an open position. It is not efficient and effective to shortlist candidates manually. Therefore Web has become more and more important platform for job recruitment for both job providers and job seekers. There are many big organisations setting up online application systems, where basic data of job profiles and application profiles are entered so that the data can be used for automated selection of candidates who have satisfied the competencies and other requirement in the job profiles. On the other hand job seekers prefer to use web portals to search for jobs that fits their competencies and other preferences. In both cases, it is a matter of matching between job demands and job offers.

When a job is advertised on the internet, a job profile will be created which specifies information about company name, location, industry, job title, job function, job type (full time or part time), length of contract, salary, demanded competencies/skills, etc. Personal profiles would typically include name, education, experience, preferred location, preferred job type, preferred salary range, competencies/skills. Among these we would need to match education, experience, location, salary and competenices/skills. The most difficult part for semantic matching is competencies.

Consider a job provider having a job position that requires candidates to have a set of programming skills, e.g. D ={Java, ER, PHP}. There are two candidates with one having programming skills S_1 ={Java, UML, PHP} and the other candidates having programming skills $S_2 = \{$Java, ER, HTML$\}$. In the literature matching between job positions and job candidates involves of measuring the distance between the concepts on an ontology hierarchy. One can use some standard competence dictionary, e.g. IEEE Reusable Competency Definition (IEEE RCD 2008) and DISCO (Müller-Riedlhuber 2009).

DISCO (Müller-Riedlhuber 2009) uses a multilingual taxonomy of competencies to provide web users with a supporting tool for filling in and translating online forms. They can be used to build competency profiles for CVs or job offers using a unique vocabulary. However, high-level competency definitions are usually not detailed or complete enough to serve as an adequate knowledge base for competency management by HR or for job recruitment. The competencies defined in DISCO, for example, are too coarse-grained to be useful for precisely describing the skills that are available or needed. In consequence, reasoning tools will not identify any suitable candidate matching the skill requirement of the job. However, human resource experts may decide that the candidate $S1$ will fit the job as he/she has the skill of using UML and should have the similar knowledge of ER because class diagram of UML can be used for modelling data. How are we going to refine the ontology hierarchy so that the system could select S_1 directly? In this paper we will present an approach of revising ontology using new domain knowledge.

3 Preliminaries

3.1 Description Logics

Description logics (DL) are a family of logic languages for representing knowledge and reasoning about it (Baader, Calvanese, McGuinness, Nardi & Patel-Schneider 2003). The knowledge of interest for some applications is represented in terms of individuals, concepts and roles, and stored in a knowledge base. Concepts stand for sets of individuals, while roles stand for relationships to other individuals. A *DL knowledge base* is a set of axioms, and usually consists of two parts: a terminological knowledge layer (called TBox) and an assertional knowledge layer (called ABox). The TBox describes the terminology in use for the application, that is, defines concepts and states additional constraints on their interpretation. The ABox describes the individuals, that is, contains assertions that relate individuals to concepts.

We briefly review syntax and semantics of description logics. Let N_C and N_R be fixed sets of concept names and of roles names. One can then build complex concept expressions out of them by using the concept constructors provided by the particular description logic being used, see Table 1. Let $\mathcal{C}$ denote the set of complex concept expressions that can be obtained by finitely many applications of these constructors. The members of N_C are called atomic concepts. We further use the empty concept $\bot$ as a shortcut for $\neg\top$, and $(\leq m).R$ is a shortcut for $\neg(\geq m + 1).R$.

EXAMPLE 3.1 For our HR application we can use atomic concepts like `Software_Engineer` or `DBA`, and roles like `hasCompetency` or `knows`. Further, we can form concept expressions like $\forall$`hasQualification.MBA` describing all individuals who have no other qualification than an MBA, or `Software_Engineer` $\sqcap$ $\exists$`hasCompetency.UML` describing all software engineers with UML skills.

Note that description logics correspond to fragments of first order logic: individuals correspond to constants, concepts to unary predicates, and roles to binary predicates. Due to restrictions set on their expressiveness the associated decision problems like satisfiability are decidable. Different description logics vary by the concept constructors that they permit. The choice of a particular description logic is usually done by balancing expressiveness against the complexity of the associated decision problems. In this paper, we use $\mathcal{ALN}$ which is among the most popular description logics. The ideas discussed in this paper are general in the sense that they can be easily tailored to other reasonably expressive description logics. Recall that $\mathcal{ALN}$ is included in $\mathcal{SHOIN}^{(\mathcal{D})}$, the description logic underlying OWL-DL.

A *subsumption axiom* is a statement of the form $C_1 \sqsubseteq C_2$ with concepts C_1, C_2 in $\mathcal{C}$. A *terminology* (or TBox) $\mathcal{T}$ is a finite set of subsumption axioms. We use the shortcut $C_1 \equiv C_2$ to denote both $C_1 \sqsubseteq C_2$ and $C_2 \sqsubseteq C_1$.

Interpretations are used to assign meaning to syntactic constructs. An interpretation $\mathcal{I}$ consists of a non-empty interpretation domain $\mathcal{O}$ and an interpretation function $\mathcal{I}(.)$, which assigns to each atomic concept a subset of $\mathcal{O}$, and to each role a binary relation on $\mathcal{O}$. The interpretation $\mathcal{I}$ can be easily extended to concept expressions in N_C. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ if $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$ holds for every subsumption axiom $C_1 \sqsubseteq C_2$ in $\mathcal{T}$. A model is finite if the interpretation domain $\mathcal{O}$ is finite. In this case, the model is also said to be an *instance* (or ABox) of $\mathcal{T}$.

A concept C_1 *subsumes* a concept C_2 if $\mathcal{I}(C_1) \subseteq \mathcal{I}(C_2)$ holds for every instance $\mathcal{I}$ of $\mathcal{T}$. We also write $\mathcal{T} \models C_1 \sqsubseteq C_2$. We use $\mathcal{T} \models C_1 \equiv C_2$ as a shortcut for $\mathcal{T} \models C_1 \sqsubseteq C_2$ and $\mathcal{T} \models C_2 \sqsubseteq C_1$, and call C_1, C_2 *equivalent*.

Example 3.2 For our HR application we may use the subsumption axiom $\texttt{Software\_Engineer} \sqsubseteq \exists\texttt{hasDegree.Software\_Engineering} \sqcup \exists\texttt{hasCompetency.Software\_Development}$ to state that software engineers must have a degree in software engineering or at least some software development skills.

The subsumption problem asks whether $\mathcal{T} \models C_1 \sqsubseteq C_2$ holds for concepts C_1, C_2 and a TBox $\mathcal{T}$. The satisfiability problem asks whether $\mathcal{T} \models C \equiv \bot$ holds for a concept C and a TBox $\mathcal{T}$. Both problems are decidable for description logics such as $\mathcal{ALN}$. For details, we refer to (Baader et al. 2003).

Example 3.3 For illustration we use a small fragment of the DISCO taxonomy (as shown in Figure 1) which gives rise to the following subsumption axioms in the TBox.

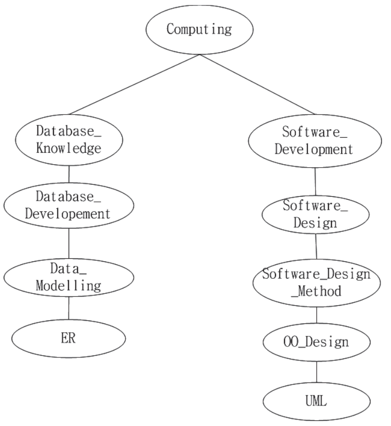

Figure 1: Competency hierarchy

$$\begin{gathered}
\texttt{Computing} \sqsubseteq \texttt{Skill}\\
\texttt{Database\_Knowledge} \sqsubseteq \texttt{Computing}\\
\texttt{Database\_Development} \sqsubseteq \texttt{Database\_Knowledge}\\
\texttt{Database\_Modelling} \sqsubseteq \texttt{Database\_Development}\\
\texttt{ER} \sqsubseteq \texttt{Database\_Modelling}\\
\texttt{Software\_Development} \sqsubseteq \texttt{Computing}\\
\texttt{Software\_Design} \sqsubseteq \texttt{Software\_Development}\\
\texttt{Software\_Design\_Methods} \sqsubseteq \texttt{Software\_Design}\\
\texttt{OO\_Design} \sqsubseteq \texttt{Software\_Design\_Methods}\\
\texttt{UML} \sqsubseteq \texttt{OO\_Design}
\end{gathered}$$

3.2 Formal Concept Analysis

Next we review basic notions from FCA (Ganter & Wille 1999). A *formal context* is a triple $\mathbb{K} := (G, M, I)$, where G is a set of individuals, M is a set of properties (or attributes), and I is a relation $I \subseteq G \times M$ that links each individual $a \in G$ to the properties satisfied by a. That is, $(a, b) \in I$ states that a has property b. A *formal concept* is a pair (A, B) such that A and B are maximal with $A \times B \subseteq I$. The set $A \subseteq G$ is called the *extent* and set $B \subseteq M$ is the *intent* of the concept. A formal concept (A_1, B_1) is a *subconcept* of a formal concept (A_2, B_2) if $A_1 \subseteq A_2$ (or equivalently $B_1 \supseteq B_2$). The concept lattice $\mathfrak{B}(\mathbb{K})$ of $\mathbb{K}$ is the set of all its FCA concepts together with the subconcept/superconcept relation. As usual, we use lattice diagrams to illustrate the ordering relation among the formal concepts in a concept lattice. The nodes of the lattice are labelled by the respective FCA concepts. For the sake of clarity, however, node labels do not show the full extent and intent of an FCA concept, but show an individual o only in the most specific FCA concept it belongs to.

Contexts can be represented as tables, with columns headed by the properties b and rows headed by individuals a. The cells of the table are marked if and only if the relations holds for the corresponding pair of individual and property.

Example 3.4 Table 2 is a formal context with four object individuals and six properties.

	b_1	b_2	b_3	b_4	b_5	b_6
a_1	×		×	×		×
a_2		×		×	×	
a_3	×		×			×
a_4			×			×

Table 2: An example of formal context

Each of the objects owns some properties. For example, object a_1 holds properties b_1, b_3, b_4 and b_6 while object a_3 holds properties b_1, b_3 and b_6. If $A = \{a_1, a_3\}$, then $B = \{b_1, b_3, b_6\}$ is the intent of the formal concept (A, B). That is, A and B is maximum with $A \times B \subseteq I$, i.e., properties $\{b_1, b_3, b_6\}$ is the maximum set of properties shared by the set of objects $\{a_1, a_3\}$ and $\{a_1, a_3\}$ is the maximum set of objects sharing the set of properties $\{b_1, b_3, b_6\}$.

Lets look at three formal concepts contained in the formal context, (A_1, B_1), (A_2, B_2), and (A_3, B_3) with $A_1 = \{a_1, a_3, a_4\}$, $B_1 = \{b_3, b_6\}$, $A_2 = \{a_1, a_3\}$, $B_2 = \{b_1, b_3, b_6\}$, $A_3 = \{a_1\}$, $B_3 = \{b_1, b_3, b_4, b_6\}$). Concepts (A_2, B_2) is a subconcept of concept (A_1, B_1) and concept (A_3, B_3) is a subset of concept (A_2, B_2). The corresponding context lattice is showing in Figure 2.

Note that the subconcept-superconcept relation is transitive. For a superconcept (A, B) all properties $b \in B$ are inherited by all its subconcepts.

The interplay between description logic and formal concept analysis has been addressed in the literature,

Constructor	Syntax	Semantics
atomic concept	A	$\mathcal{I}(A) \subseteq \mathcal{O}$
role	R	$\mathcal{I} \subseteq \mathcal{O} \times \mathcal{O}$
universal concept	$\top$	$\mathcal{I}(\top) = \mathcal{O}$
negation	$\neg C$	$\mathcal{I}(\neg C) = \mathcal{O} - \mathcal{I}(C)$
conjunction	$C_1 \sqcap C_2$	$\mathcal{I}(C_1 \sqcap C_2) = \mathcal{I}(C_1) \cap \mathcal{I}(C_2)$
disjunction	$C_1 \sqcup C_2$	$\mathcal{I}(C_1 \sqcup C_2) = \mathcal{I}(C_1) \cup \mathcal{I}(C_2)$
existential	$\exists R.C$	$\mathcal{I}(\exists R.C) = \{x \in \mathcal{O} \mid (x,y) \in \mathcal{I}(R) \text{ for some } y \in \mathcal{I}(C)\}$
universal	$\forall R.C$	$\mathcal{I}(\forall R.C) = \{x \in \mathcal{O} \mid (x,y) \in \mathcal{I}(R) \text{ for all } y \in \mathcal{I}(C)\}$
lower bound	$(\geq m.R)$	$\mathcal{I}(\geq m.R) = \{x \in \mathcal{O} \mid \#\{y \in \mathcal{O} \mid (x,y) \in \mathcal{I}(R)\} \geq m\}$

Table 1: Syntax and semantics of $\mathcal{ALN}$.

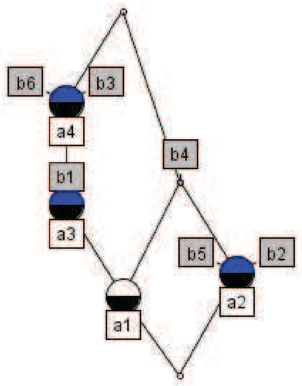

Figure 2: Concept Lattice for the context of Table 2

see (Sertkaya 2011) for a recent survey. While both study concepts, there are differences between the approaches. Most notable, DL and FCA use the term "concept" with different meanings, hence we will always speak of FCA concepts or DL concepts in the rest of the paper. FCA starts with the properties, and then inspects the complete extensional description of an application domain to conclude the concepts of this specific domain. DL, on the other hand, starts with an intensional definition of the concepts that is given independently of a specific domain (Baader & Sertkaya 2004). FCA uses atomic properties to classify individuals by the properties that they observe. DL uses atomic concepts and builds complex concept descriptions from them (and the roles). As pointed out by several authors, DL concepts correspond to FCA properties (Sertkaya 2011). So when using FCA to build a concept lattice one should choose M as the set N_C of atomic DL concepts, and define I such that $(a, b) \in I$ whenever individual a belongs to DL concept b in some ABox under inspection.

4 Revise Ontology Using FCA

We found it convenient to view a TBox as an ontology hierarchy. Let $C_1, \ldots, C_n$ be the non-trivial concepts of interest for which subsumption axioms are recorded in a TBox $\mathcal{T}$. Let $M = \{C_1, \ldots, C_n\}$ and let $\sqsubseteq_{\mathcal{H}}$ be the partial order on M induced by subsumption axioms in $\mathcal{T}$ (that is, the reflexive and transitive closure of $\mathcal{T}$ projected onto M). We call $\mathcal{H} = (M, \sqsubseteq_{\mathcal{H}})$ an ontology hierarchy of $\mathcal{T}$. Figure 1 shows the diagram of an ontology hierarchy for the TBox in Example 3.3.

Ontologies need to be maintained to reflect the changing of the real world. For example GML emerged as a new standard for exchanging geographical data over the internet. This change should be reflected by the competency ontology so that the jobs requiring GML skills can be captured. Therefore, competency ontology should be revised when new knowledge is captured, e.g. information of new advised jobs or input of human resource experts. From information of new jobs we get some new objects that can be used to created FCA matrix. In this section we first present an ontology revision process. Then we will show how FCA can be used to revise the ontology hierarchy of a DL.

4.1 Ontology Revision Process

To maintain and revise existing ontology hierarchies we take an inputs an existing ontology hierarchy and a data set of a domain and output a revised ontology. The following describes steps of the process which is depicted in Figure 3.

1. Create representative ABox for a given ontology hierarchy by associating each of the concepts on the hierarchy with an object g_i,
2. Generate a representative FCA context $\mathbb{K} := (G, M, I)$ for a given ontology hierarchy $\mathcal{H}$ as a master context,
3. Analyse new data sets and generate a FCA context $\mathbb{K}' := (G', M', I')$,
4. Update the intent of the master FCA context $\mathbb{K}$ by setting $M_{new} = M \cup M'$,
5. Insert new objects into the master context to get revised formal context $\mathbb{K}_{new}$, merging tuples when they share a concept with any existing tuples,
6. Generate new FCA lattice from new FCA context $\mathbb{K}_{new}$,
7. Abstract new ontology hierarchy from the new FCA lattice $\mathfrak{B}(\mathbb{K}_{new})$.

We will show later in Section 4.2 that the representative context created provides the same implication as subsumption relations given by the ontology hierarchy. Also, we will show how to use the above process with some examples in the following subsections.

4.2 Generate Representative FCA Context

In the following let $\mathcal{H}$ be a given ontology hierarchy, and $M = \{C_1, \ldots, C_n\}$ the set of concepts defined in $\mathcal{H}$ and $\sqsubseteq_{\mathcal{H}}$ the subsumption relation of $\mathcal{H}$. We can then derive a representative ABox $\mathcal{S}^{\mathcal{H}}$ for $\mathcal{H}$ as follows: choose an n-element subset $G = \{g_1, \ldots, g_n\}$ of the interpretation domain $\mathcal{O}$, and define the interpretation function $\mathcal{I}$ by $\mathcal{I}(C) = \{g_i \in G : C \sqsubseteq_{\mathcal{H}} C_i\}$ for $C \in M$.

It is easy to check that the interpretation $\mathcal{I}$ is indeed a model of $\mathcal{H}$. The ABox $\mathcal{S}^{\mathcal{H}}$ can easily translated into a formal context $\mathbb{K}^{\mathcal{H}} = (G, M, I)$ with $(g, C) \in I$ just when $g \in \mathcal{I}(C)$ holds for the concept $C \in M$. In the FCA matrix of $\mathbb{K}^{\mathcal{H}}$ we have a

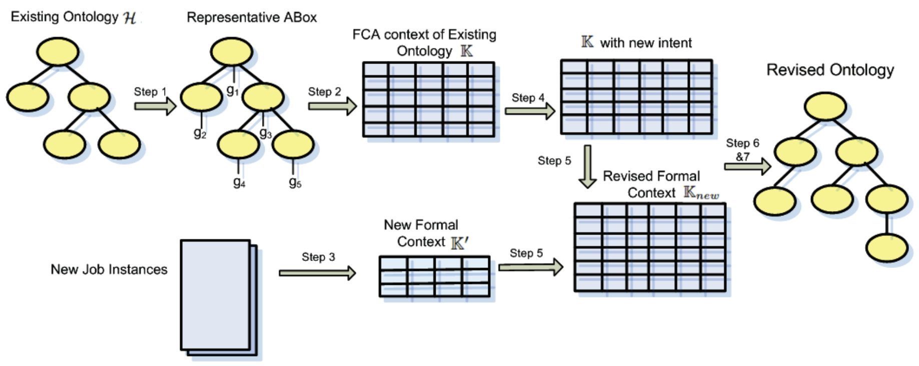
Existing Ontology $\mathcal{H}$
Representative ABox
FCA context of Existing Ontology $\mathbb{K}$
$\mathbb{K}$ with new intent
Revised Ontology
Step 1
Step 2
Step 4
Step 5
Step 6 &7
Revised Formal Context $\mathbb{K}_{new}$
New Formal Context $\mathbb{K}'$
New Job Instances
Step 3
Step 5
g1
g2
g3
g4
g5

Figure 3: FCA based Ontology Revision Process

row for every $g \in G$ such that there is a 1 in column C if and only if g satisfies concept C. We call $\mathbb{K}^{\mathcal{H}}$ a *representative FCA context* for $\mathcal{H}$. Note that the properties of $\mathbb{K}^{\mathcal{H}}$ are just the concepts of $\mathcal{H}$.

For $g_i \in G$ let $A_i = \{g_i\}$ and $B_i = \{C \in M : C_i \sqsubseteq_{\mathcal{H}} C\}$. It is easy to see that the pair (A_i, B_i) forms an FCA concept of $\mathbb{K}^{\mathcal{H}}$. We call it the FCA representative of the concept C_i. By definition, (A_i, B_i) is a superconcept of (A_j, B_j) just when $B_i \subseteq B_j$, and this holds if and only if $C_i \sqsubseteq C_j$. We record this as follows:

Proposition 1 The ontology hierarchy $\mathcal{H} = (M, \sqsubseteq_{\mathcal{H}})$ is isomorphic to the subconcept relation on the set of FCA concepts (A_i, B_i) with $i = 1, \ldots, n$.

Example 4.1 Following the process in 4.1, in Step 1 we create a representative ABox or a given ontology. For the example ontology hierarchy in Example 3.3 we associate an object for each concept in the hierarchy, say job descriptions $J_1, \ldots, J_{10}$, so that the following assertions are satisfied:

$$\begin{gathered}\text{Computing}(J_1)\\ \text{Database\_Knowledge}(J_2)\\ \text{Database\_Development}(J_3)\\ \text{Database\_Modelling}(J_4)\\ \text{ER}(J_5)\\ \text{Software\_Development}(J_6)\\ \text{Software\_Design}(J_7)\\ \text{Software\_Design\_Method}(J_8)\\ \text{OO\_Design}(J_9)\\ \text{UML}(J_{10})\end{gathered}$$

We can easily extend this to a representative ABox of the ontology hierarchy, and construct a representative FCA context for it, see Figure 4. Note that each job description J_i holds the DL concept that it associates with and all the DL concepts above it on the ontology hierarchy. For example, J_3 holds the DL concept Database_Development and also the DL concepts above it, i.e. Database_Knowledge and Computing.

The corresponding FCA lattice is shown in Figure 5. The ontology hierarchy $\mathcal{H}$ given in Example 3.3 is isomorphic to the subconcept relation induced by the FCA representatives of the concepts in $\mathcal{H}$.

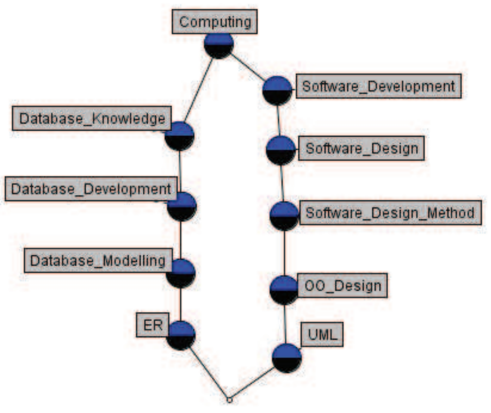

Figure 5: Skill Lattice

4.3 Update Ontology Hierarchies

In this section we present an algorithm for updating existing master ontology hierarchies. Our approach makes use of FCA contexts. Assume that the master ontology and the next generated context are from the same domain. There are some overlaps of attributes.

Example 4.2 Consider the ontology in Example 4.1 and assume that a domain expert concludes that if a candidate knows UML then he/she has the skill of the database modelling because he/she knows Class Diagram. So a FCA context is created with one object, J_{11}, as shown in Figure 6.

According to the ontology revision process in Figure 3 we merge the two contexts, Master Skill Context and New Skill Context, into one as in Figure 7.

Note that when we are merging contexts we first check if there is any existing object sharing an ontology concept (or FCA attribute). For example we check if there is an object in the original matrix has the attribute UML. Because there is one we then merge the two objects into one J_{10} and allocate all the attributes in J_{10} and J_{11} to it.

Using the updated context we can generate a FCA lattice as in Figure 8.

Note that if an object satisfies DL concept Database_Knowledge it also satisfies DL concept Computing because Database_Knowledge $\sqsubseteq$ Computing.

As mentioned in Section 1, the motivation of maintaining ontology is to provide better matching results in the process of human resource recruitment. In the literature there are many approaches using ontology hierarchies to calculate distance or similarities between ontology concepts (Bizer et al. 2005, Fazel-Zarandi et al. 2011, Janowicz, Raubal & Kuhn 2011).

	Computing	Database_Knowledg	Database_Development	Database_Modelling	ER	Software_Develop	Software_Design	Software_Design_Method	OO_Design	UML
J1	1									
J2	1	1								
J3	1	1	1							
J4	1	1	1	1						
J5	1	1	1	1	1					
J6	1					1				
J7	1					1	1			
J8	1					1	1	1		
J9	1					1	1	1	1	
J10	1					1	1	1	1	1

Figure 4: Master Skill Context

Algorithm 1 (Generating of a representative context of FCA lattice)

```
Input:   a FCA master context $\mathbb{K} := \{G, M, I\}$ generated from a given $\mathcal{H}$
         a new FCA context $\mathbb{K} := (G', M', I')$ /*a context of set new objects
Output:  a revised FCA context $\mathbb{K}_{new} := (G_{new}, M_{new}, I_{new})$
Method:  find new attributes $M'_{new} = M' \setminus M$
             extend the intent of the master context $M$ to $M_{new} = M \cup M'_{new}$
         endfor
         for each object $g_i \in G'$ with $B' \subseteq M' \wedge g_i I' B'_i$ do
             if there exist $g_j \in G$ such that $g_j I B_j \wedge B_j \subseteq M \wedge B'_i \cap B_j \neq \emptyset$ then
                update $g_i$ and set $g_i I_{new} (B'_i \cup B_j)$
             endif
             insert $g_i$ into $\mathbb{K}$
         endfor
```

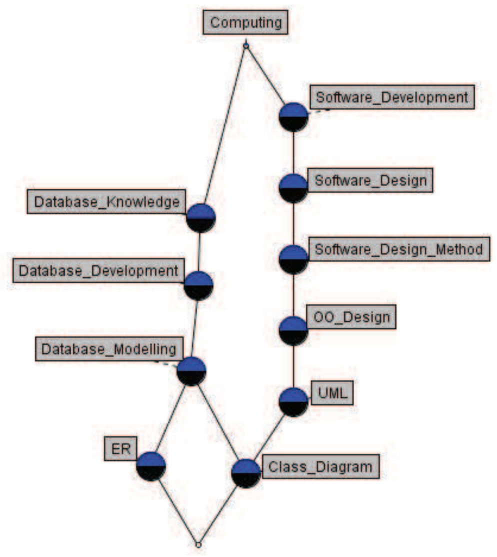

Figure 8: Revised Skill Lattice

It calculates the distances to the common parents, with some of them using different weights for different levels and some not considering differences at all. All approaches of using distances of attributes on a hierarchy for matching consider that the closer the pair of attributes to the common parents the better of matching they are.

In Figure 8 Class_Diagram has two parents, UML and Database_Modelling. So Class_Diagram and ER are siblings with common parent Database_Modeling. Without considering the weights of edges, the distance from Class_Diagram to Database_Modelling $Distance_{\mathcal{T}_2}$(Class_Diagram, Database_Modelling) = 1. Obviously the distance is shorter than the distance between UML and Database_Modelling in the previous ontology hierarchy $\mathcal{T}_1$, i.e. $Distance_{\mathcal{T}_1}$(UML, Database_Modelling) = 4 (see Figure1).

As we see in Example 4.2, quality of ontology hierarchies used for matching affects the results of competence matching. Our approach of updating ontology hierarchy can lead to better matching results for the motivation example in Section 2, i.e. S_1 could be selected using the revised ontology hierarchy.

5 Related Work and Discussion

A survey conducted in (Sertkaya 2011) groups the works on bridging the gap between FCA and DLs into two categories, with some work on enriching the language of FCA by borrowing constructors from DL languages (Prediger & Stumme 1999) while some other work on employing FCA methods to solve problem encountered in knowledge representation with DLs.

FCA has been used in building subsumption hierarchies. Most works in the literature apply FCA to build subsumption hierarchies from scratch. In (Baader & Sertkaya 2004), FCA is applied to build subsumption hierarchies with given set of description logic. (Prediger & Stumme 1999) extracts data from relational databases to construct a formal context by using conceptual scales. DLs and attribute exploration is used to specify which attributes cannot occur together and DL reasoner is used during the attribute exploration process as an expert to answer the implication questions and provide counterexample whenever the implication does not hold. In (Baader 1995) FCA is used to compute an extended subsumption hierarchy by building a formal context of a give set of DL concepts where attributes of the formal context are the defined DL concepts. In (Baader & Molitor 2000), FCA is used in bottom-up construction of DL knowledge bases. Attribute exploration is used to compute the subsumption hierarchy of all least common subsumers of a given set of concepts.

FCA has also been used for merging ontology hierarchies in (Bendaoud, Napoli & Toussaint 2008, Cur & Jeansoulin 2008, Stumme & Maedche 2001). However all of them assume that sample data sets are given for any give ontologies. Merging of ontology is then performed by merging the existing data sets. However, very often we can obtain an ontology hier-

	Computing	Database_Knowledge	Database_Development	Database_Modelling	ER	Software_Development	Software_Design	Software_Design_Method	OO_Design	UML	Class_Diagram
J11	1	1	1	1							1

Figure 6: New Skill Context

	Computing	Database_Knowledge	Database_Development	Database_Modelling	ER	Software_Development	Software_Design	Software_Design_Method	OO_Design	UML	Class_Diagram
J1	1										
J2	1	1									
J3	1	1	1								
J4	1	1	1	1	1						
J5	1	1	1	1	1						
J6	1					1					
J7	1					1	1				
J8	1					1	1	1			
J9	1					1	1	1	1		
J10	1	1	1	1		1	1	1	1	1	1

Figure 7: Revised Context Skill

archy but without any corresponding data sets. In this case, the approaches of merging ontology using existing data sets cannot be applied. Our approach of maintaining existing ontology hierarchies does not require an matching data sets and therefore can be applied in all the situations of ontology maintenance. Further, for a given ontology hierarchy our approach can generate a representative data sets, which is of small size, if it is not minimal. The following step of updating ontology using the representative data sets can be performed efficiently as only small number of objects need to be merged. Furthermore, the aim of maintaining and revising existing ontology is to improve semantic searching result. Our approach takes input new knowledge from domain experts and uses FCA to insert new relations and new DL concepts in the existing ontology hierarchies. With the maintaining and revising of ontology hierarchies the results of semantic searching and matching can be improved.

6 Conclusion

In this paper we have proposed an approach for maintaining ontology hierarchies using formal concept analysis. We have proposed a process for updating existing ontology hierarchies together with an algorithm for updating ontology hierarchies. This approach is efficient for updating existing ontologies with new subsumption relations or new concepts. We have shown that the approach can indeed improve existing ontologies so that by using it matching results can be improved.

References

Baader, F. (1995), Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology, *in* 'Proc. Intl. KRUSE Symposium', pp. 168–178.

Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D. & Patel-Schneider, P. F., eds (2003), *The description logic handbook: theory, implementation, and applications*, Cambridge University Press.

Baader, F. & Molitor, R. (2000), Building and structuring description logic knowledge bases using least common subsumers and concept analysis, *in* '8th International Conference on Conceptual Structures, (ICCS 2000)', pp. 292–305.

Baader, F. & Sertkaya, B. (2004), Applying formal concept analysis to description logics, *in* P. Eklund, ed., 'Concept Lattices', Vol. 2961 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelberg, pp. 593–594.

Bendaoud, R., Napoli, A. & Toussaint, Y. (2008), Formal concept analysis: A unified framework for building and refining ontologies, *in* A. Gangemi & J. Euzenat, eds, '16th International Conference on Knowledge Engineering and Knowledge Management - EKAW 2008', Vol. 5268 of *Lecture Notes in Artificial Intelligence*, Springer Berlin / Heidelberg, Acitrezza, Catania, Italy, pp. 156–171.

Bizer, C., Heese, R., Mochol, M., Oldakowski, R., Tolksdorf, R., Berlin, F. U. & Eckstein, R. (2005), The impact of semantic web technologies on job recruitment processes, *in* 'In Proceedings of the 7th International Conference Wirtschaftsinformatik', pp. 1367–1383.

Cimiano, P., Hotho, A. & Staab, S. (2005), 'Learning concept hierarchies from text corpora using formal concept analysis', *Journal of Artificial Intelligence research* **24**, 305–339.

Colucci, S., Noia, T. D., Sciascio, E. D., Donini, F. M., Mongiello, M. & Mottola, M. (2003), 'A formal approach to ontology-based semantic match of skills descriptions', *J. of Universal Computer Science, Special issue on Skills Management* **9**, 1437–1454.

Cur, O. & Jeansoulin, R. (2008), An fca-based solution for ontology mediation., *in* R. Elmasri, M. Doerr, M. Brochhausen & H. Han, eds, 'ONISW', ACM, pp. 39–46.

Fazel-Zarandi, M., Devlin, H. J., Huang, Y. & Contractor, N. (2011), Expert recommendation based on social drivers, social network analysis, and semantic data representation, *in* 'Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems', HetRec '11, ACM, New York, NY, USA, pp. 41–48.

Ganter, B. & Wille, R. (1999), *Formal Concept Analysis*, Springer.

Haav, H.-M. (2004), A semi-automatic method to ontology design by using fca, *in* 'Proceedings of the CLA 2004 International Workshop on Concept Lattices and their Applications'.

Herder, E. (2009), ‘Competence matching tool’. http://dspace.ou.nl/bitstream/1820/2282/1/competencematchingtool.pdf.

IEEE RCD (2008), ‘Ieee standard for learning technology - data model for reusable competency definitions’.

Janowicz, K., Raubal, M. & Kuhn, W. (2011), ‘The semantics of similarity in geographic information retrieval’, *Journal of Spatial Information Science* **2**(2), 29–57.

Jia, H., Newman, J. & Tianfield, H. (2007), A new formal concept analysis based learning approach to ontology building, *in* ‘Proceedings of the 2nd International Conference on Metadata and Semantics Research (MTSR 2007)’, pp. 433–444.

Lammari, N. & Métais, E. (2004), ‘Building and maintaining ontologies: a set of algorithms’, *Data and Knowledge Engineering* **48**(2), 155 – 176. ¡ce:title¿Applications of Natural Language to Information Systems (NLDB) 2002¡/ce:title¿.

Müller-Riedlhuber, H. (2009), The european dictionary of skills and competences (disco): an example of usage scenarios for ontologies, *in* ‘Proc. 5th International Conference on Semantic Systems, ISEMANTICS’, JUCS Conference Proceedings Series, pp. 467–479.

Prediger, S. & Stumme, G. (1999), Theory-driven logical scaling, *in* ‘Proc. 6th Intl. Workshop Knowledge Representation Meets Databases, Heidelberg. CEUR Workshop Proc’, pp. 46–49.

Sánchez, F. G., Martínez-Béjar, R., Contreras, L., Fernández-Breis, J. T. & Nieves, D. C. (2006), ‘An ontology-based intelligent system for recruitment’, *Expert Systems with Applications* **31**(2), 248–263.

Sertkaya, B. (2011), ‘A survey on how description logic ontologies benefit from formal concept analysis’, *CoRR* **abs/1107.2822**.

Stumme, G. & Maedche, A. (2001), Fca–merge: Bottom-up merging of ontologies, *in* ‘Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI’2001)’, Morgen Kaufmann, pp. 225–234.

Wille, R. (1982), Restructuring lattice theory: an approach based on hierarchies of concepts., *in* ‘Rival, I. (ed.): Ordered Sets’, Boston, pp. 445–470. seminal publication on formal concept analysis.