Completing Description Logic Knowledge Bases using Formal Concept Analysis*

Franz Baader,[1] **Bernhard Ganter,**[1] **Barış Sertkaya,**[1] **and Ulrike Sattler**[2]
[1]TU Dresden, Germany and [2]The University of Manchester, UK

Abstract

We propose an approach for extending both the terminological and the assertional part of a Description Logic knowledge base by using information provided by the knowledge base and by a domain expert. The use of techniques from Formal Concept Analysis ensures that, on the one hand, the interaction with the expert is kept to a minimum, and, on the other hand, we can show that the extended knowledge base is complete in a certain, well-defined sense.

1 Introduction

Description Logics (DLs) [Baader *et al.*, 2003] are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, databases, and bio-medical ontologies, but their most notable success so far is due to the fact that DLs provide the logical underpinning of OWL, the standard ontology language for the semantic web [Horrocks *et al.*, 2003]. As a consequence of this standardization, several ontology editors support OWL [Knublauch *et al.*, 2004; Oberle *et al.*, 2004; Kalyanpur *et al.*, 2006a], and ontologies written in OWL are employed in more and more applications. As the size of these ontologies grows, tools that support improving their quality become more important. The tools available until now use DL reasoning to detect inconsistencies and to infer consequences, i.e., implicit knowledge that can be deduced from the explicitly represented knowledge. There are also promising approaches that allow to pinpoint the reasons for inconsistencies and for certain consequences, and that help the ontology engineer to resolve inconsistencies and to remove unwanted consequences [Schlobach and Cornet, 2003; Kalyanpur *et al.*, 2006b]. These approaches address the quality dimension of *soundness* of an ontology, both within itself (consistency) and w.r.t. the intended application domain (no unwanted consequences). In the present paper, we are concerned with a different quality dimension: *completeness*. We provide a basis for formally well-founded techniques and tools that support the ontology engineer in checking whether an ontology contains all the relevant information about the application domain, and to extend the ontology appropriately if this is not the case.

A DL knowledge base (nowadays often called ontology) usually consists of two parts, the terminological part (TBox), which defines concepts and also states additional constraints (so-called general concept inclusions, GCIs) on the interpretation of these concepts, and the assertional part (ABox), which describes individuals and their relationship to each other and to concepts. Given an application domain and a DL knowledge base (KB) describing it, we can ask whether the KB contains all the relevant information[1] about the domain:

- Are all the relevant constraints that hold between concepts in the domain captured by the TBox?
- Are all the relevant individuals existing in the domain represented in the ABox?

As an example, consider the OWL ontology for human protein phosphatases that has been described and used in [Wolstencroft *et al.*, 2005]. This ontology was developed based on information from peer-reviewed publications. The human protein phosphatase family has been well characterised experimentally, and detailed knowledge about different classes of such proteins is available. This knowledge is represented in the terminological part of the ontology. Moreover, a large set of human phosphatases has been identified and documented by expert biologists. These are described as individuals in the assertional part of the ontology. One can now ask whether the information about protein phosphatases contained in this ontology is complete. Are all the relationships that hold among the introduced classes of phosphatases captured by the constraints in the TBox, or are there relationships that hold in the domain, but do not follow from the TBox? Are all possible kinds of human protein phosphatases represented by individuals in the ABox, or are there phosphatases that have not yet been included in the ontology or even not yet been identified?

Such questions cannot be answered by an automated tool alone. Clearly, to check whether a given relationship between

*Supported by DFG (GRK 334/3) and the EU (IST-2005-7603 FET project TONES and NoE 507505 Semantic Mining).

[1]The notion of "relevant information" must, of course, be formalized appropriately for this problem to be addressed algorithmically.

concepts—which does not follow from the TBox—holds in the domain, one needs to ask a domain expert, and the same is true for questions regarding the existence of individuals not described in the ABox. The rôle of the automated tool is to ensure that the expert is asked as few questions as possible; in particular, she should not be asked trivial questions, i.e., questions that could actually be answered based on the represented knowledge. In the above example, answering a non-trivial question regarding human protein phosphatases may require the biologist to study the relevant literature, query existing protein databases, or even to carry out new experiments. Thus, the expert may be prompted to acquire new biological knowledge.

Attribute exploration is an approach developed in Formal Concept Analysis (FCA) [Ganter and Wille, 1999] that can be used to acquire knowledge about an application domain by querying an expert. One of the earliest applications of this approach is described in [Wille, 1982], where the domain is lattice theory, and the goal of the exploration process is to find, on the one hand, all valid relationships between properties of lattices (like being distributive), and, on the other hand, to find counterexamples to all the relationships that do not hold. To answer a query whether a certain relationship holds, the lattice theory expert must either confirm the relationship (by using results from the literature or carrying out a new proof for this fact), or give a counterexample (again, by either finding one in the literature or constructing a new one).

Although this sounds very similar to what is needed in our context, we cannot directly use this approach. The main reason is the open-world semantics of description logic knowledge bases. Consider an individual i from an ABox $\mathcal{A}$ and a concept C occurring in a TBox $\mathcal{T}$. If we cannot deduce from the TBox $\mathcal{T}$ and $\mathcal{A}$ that i is an instance of C, then we do not assume that i does not belong to C. Instead, we only accept this as a consequence if $\mathcal{T}$ and $\mathcal{A}$ imply that i is an instance of $\neg C$. Thus, our knowledge about the relationships between individuals and concepts is incomplete: if $\mathcal{T}$ and $\mathcal{A}$ imply neither $C(i)$ nor $\neg C(i)$, then we do not know the relationship between i and C. In contrast, classical FCA and attribute exploration assume that the knowledge about individuals is complete: the basic datastructure is that of a formal context, i.e., a crosstable between individuals and properties. A cross says that the property holds, and the absence of a cross is interpreted as saying that the property does not hold.

There has been some work on how to extend FCA and attribute exploration from complete knowledge to the case of partial knowledge [Obiedkov, 2002; Burmeister and Holzer, 2005]. However, this work is based on assumptions that are different from ours. In particular, it assumes that the expert cannot answer all queries and, as a consequence, the knowledge obtained after the exploration process may still be incomplete and the relationships between concepts that are produced in the end fall into two categories: relationships that are valid no matter how the incomplete part of the knowledge is completed, and relationships that are valid only in some completions of the incomplete part of the knowledge. In contrast, our intention is to complete the KB, i.e., in the end we want to have complete knowledge about these relationships. What may be incomplete is the description of individuals used during the exploration process.

In the next section, we introduce our variant of FCA that can deal with partial contexts, and describe an attribute exploration procedure that works with partial contexts. In Section 3, we show how a DL knowledge base gives rise to a partial context, define the notion of a completion of a knowledge base w.r.t. a fixed model, and show that the attribute exploration algorithm developed in the previous section can be used to complete a knowledge base. In Section 4 we describe ongoing and future work. This paper is accompanied by a technical report [Baader *et al.*, 2006] containing more details and in particular full proofs of our results.

2 Exploring Partial Contexts

In this section, we extend the classical approach to Formal Concept Analysis (FCA), as described in detail in [Ganter and Wille, 1999], to the case of individuals (called objects in FCA) that have only a partial description in the sense that, for some properties (called attributes in FCA), it is not known whether they are satisfied by the individual or not. The connection between this approach and the classical approach is explained in [Baader *et al.*, 2006]. In the following, we assume that we have a finite set M of attributes and a (possibly infinite) set of objects.

Definition 2.1 *A* partial object description (pod) *is a tuple (A, S) where $A, S \subseteq M$ are such that $A \cap S = \varnothing$. We call such a pod a* full object description (fod) *if $A \cup S = M$. A set of pods is called a* partial context *and a set of fods a* full context.

Intuitively, the pod (A, S) says that the object it describes satisfies all attributes from A and does not satisfy any attribute from S. For the attributes not contained in $A \cup S$, nothing is known w.r.t. this object. A partial context can be extended by either adding new pods or by extending existing pods.

Definition 2.2 *We say that the pod (A', S')* extends *the pod (A, S), and write this as $(A, S) \leq (A', S')$, if $A \subseteq A'$ and $S \subseteq S'$. Similarly, we say that the partial context $\mathcal{K}'$* extends *the partial context $\mathcal{K}$, and write this as $\mathcal{K} \leq \mathcal{K}'$, if every pod in $\mathcal{K}$ is extended by some pod in $\mathcal{K}'$. If $\overline{\mathcal{K}}$ is a full context and $\mathcal{K} \leq \overline{\mathcal{K}}$, then $\overline{\mathcal{K}}$ is called a* realizer *of $\mathcal{K}$. If $(\overline{A}, \overline{S})$ is a fod and $(A, S) \leq (\overline{A}, \overline{S})$, then we also say that $(\overline{A}, \overline{S})$* realizes *$(A, S)$.*

Next, we introduce the notion of an implication between attributes which formalizes the informal notion "relationship between properties" used in the introduction.

Definition 2.3 *An* implication *is of the form $L \rightarrow R$ where $L, R \subseteq M$. This implication is* refuted *by the pod (A, S) if $L \subseteq A$ and $R \cap S \neq \varnothing$. It is* refuted *by the partial context $\mathcal{K}$ if it is refuted by at least one element of $\mathcal{K}$. The set of implications that are not refuted by a given partial context $\mathcal{K}$ is denoted by $Imp(\mathcal{K})$. The set of all fods that do not refute a given set of implications $\mathcal{L}$ is denoted by $Mod(\mathcal{L})$.*

For a set of implications $\mathcal{L}$ and a set $P \subseteq M$, the *implicational closure* of P with respect to $\mathcal{L}$, denoted by $\mathcal{L}(P)$, is the smallest subset Q of M such that

- $P \subseteq Q$, and

- $L \to R \in \mathcal{L}$ and $L \subseteq Q$ imply $R \subseteq Q$.

A set $P \subseteq M$ is called $\mathcal{L}$*-closed* if $\mathcal{L}(P) = P$.

Definition 2.4 *The implication $L \to R$ is said to* follow *from a set $\mathcal{J}$ of implications if $R \subseteq \mathcal{J}(L)$. The set of implications $\mathcal{J}$ is called* complete *for a set of implications $\mathcal{L}$ if every implication in $\mathcal{L}$ follows from $\mathcal{J}$. It is called* sound *for $\mathcal{L}$ if every implication that follows from $\mathcal{J}$ is contained in $\mathcal{L}$. A set of implications $\mathcal{J}$ is called a* base *for a set of implications $\mathcal{L}$ if it is both sound and complete for $\mathcal{L}$, and no strict subset of $\mathcal{J}$ satisfies this property.*

The following is a trivial fact regarding the connection between partial contexts and the implications they do not refute, but it will turn out to be crucial for our attribute exploration algorithm.

Proposition 2.5 *For a given set $P \subseteq M$ and a partial context $\mathcal{K}$, $\mathcal{K}(P) := M \setminus \bigcup\{S \mid (A, S) \in \mathcal{K}, P \subseteq A\}$ is the largest subset of M such that $P \to \mathcal{K}(P)$ is not refuted by $\mathcal{K}$.*

Attribute exploration with partial contexts

The classical attribute exploration algorithm of FCA assumes that there is a domain expert that can answer questions regarding the validity of implications in the application domain. Accordingly, our approach requires an expert that can decide whether an implication is refuted in the application domain or not. In contrast to existing work on extending FCA to the case of partial knowledge [Obiedkov, 2002; Burmeister and Holzer, 2005], we do *not* assume that the expert has only partial knowledge and thus cannot answer all implication questions.

To be more precise, we consider the following setting. We are given an initial (possibly empty) partial context $\mathcal{K}$, an initially empty set of implications $\mathcal{L}$, and a full context $\overline{\mathcal{K}}$ that is a realizer of $\mathcal{K}$. The expert answers implication questions "$L \to R$?" w.r.t. the full context $\overline{\mathcal{K}}$. More precisely, if the answer is "yes," then $\overline{\mathcal{K}}$ does not refute $L \to R$. The implication $L \to R$ is then added to $\mathcal{L}$. Otherwise, the expert extends the current context $\mathcal{K}$ such that the extended context refutes $L \to R$ and still has $\overline{\mathcal{K}}$ as a realizer. Consequently, the following invariant will be satisfied by $\mathcal{K}, \overline{\mathcal{K}}, \mathcal{L}$:

$$\mathcal{K} \leq \overline{\mathcal{K}} \subseteq \mathit{Mod}(\mathcal{L}).$$

Our aim is to enrich $\mathcal{K}$ and $\mathcal{L}$ such that eventually $\mathcal{L}$ is not only sound, but also complete for $\mathit{Imp}(\overline{\mathcal{K}})$, and $\mathcal{K}$ refutes all other implications (i.e., all the implications refuted by $\overline{\mathcal{K}}$). As in the classical case, we want to do this by asking as few as possible questions to the expert.

Definition 2.6 *Let $\mathcal{L}$ be a set of implications and $\mathcal{K}$ a partial context. An implication is called* undecided *w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it neither follows from $\mathcal{L}$ nor is refuted by $\mathcal{K}$. It is* decided *w.r.t. $\mathcal{K}$ and $\mathcal{L}$ if it is not undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$.*

In principle, our attribute exploration algorithm tries to decide each undecided implications by either adding it to $\mathcal{L}$ or extending $\mathcal{K}$ such that it refutes the implication. If all implications are decided, then our goal is achieved.

Proposition 2.7 *Assume that $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq \mathit{Mod}(\mathcal{L})$ and that all implications are decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. Then $\mathcal{L}$ is complete for $\mathit{Imp}(\overline{\mathcal{K}})$ and $\mathcal{K}$ refutes all implications not belonging to $\mathit{Imp}(\overline{\mathcal{K}})$.*

How can we find—and let the expert decide—all undecided implications without considering all implications? The following proposition motivates why it is sufficient to consider implications whose left-hand sides are $\mathcal{L}$-closed.

Proposition 2.8 *Let $\mathcal{L}$ be a set of implications and $L \to R$ an implication. Then, $L \to R$ follows from $\mathcal{L}$ iff $\mathcal{L}(L) \to R$ follows from $\mathcal{L}$.*

Concerning right-hand sides, Proposition 2.5 says that the largest right-hand side R such that $L \to R$ is not refuted by $\mathcal{K}$ is $R = \mathcal{K}(L)$. Putting these two observations together, we only need to consider implications of the form $L \to \mathcal{K}(L)$ where L is $\mathcal{L}$-closed. In order to enumerate all left-hand sides, we can thus use the well-known approach from FCA for enumerating closed sets in the lectic order [Ganter and Wille, 1999].

Definition 2.9 *Assume that $M = \{m_1, \ldots, m_n\}$ and fix some linear order $m_1 < m_2 < \cdots m_n$ on M. The* lectic order $<$ *is defined as follows: for $m_i \in M$ and $A, B \subseteq M$ we define*

$$A <_i B \text{ iff } m_i \in B \setminus A \text{ and } A \cap \{m_1, \ldots, m_{i-1}\} = B \cap \{m_1, \ldots, m_{i-1}\}.$$

The order $<$ is the union of the orders $<_i$.

Obviously, $<$ extends the strict subset order, and thus $\varnothing$ is the smallest and M the largest set w.r.t. $<$.

Proposition 2.10 *Given a set of implications $\mathcal{L}$ and an $\mathcal{L}$-closed set $A \subsetneq M$, the next $\mathcal{L}$-closed set following A in the lectic order is $\mathcal{L}((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ where j is maximal such that $A <_j \mathcal{L}((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$.*

If an implication is added because the expert has stated that it holds in $\overline{\mathcal{K}}$, then we can extend the current context $\mathcal{K}$ by closing the first component of every pod in $\mathcal{K}$ w.r.t. the new set of implications $\mathcal{L}$. In fact, $\mathcal{L} \subseteq \mathit{Imp}(\overline{\mathcal{K}})$ makes sure that the extended context is still realized by $\overline{\mathcal{K}}$. To allow for this and possible other ways of extending the partial context, the formulation of the algorithm just says that, in case an implication is added, the partial context can also be extended. Whenever an implication is not accepted by the expert, $\mathcal{K}$ will be extended to a context that refutes the implication and still has $\overline{\mathcal{K}}$ as a realizer.

Based on these considerations, our attribute exploration algorithm for partial contexts is described in Algorithm 1. It takes a partial context $\mathcal{K}_0$ and a set M of attributes to be explored, and interacts with an expert who has information about a full context $\overline{\mathcal{K}}$ that realizes $\mathcal{K}_0$. It goes through the set of implications over M in the lectic order of their left hand sides and, in case they are undecided in the current partial context, asks the expert to decide them. In case an implication holds in $\overline{\mathcal{K}}$, (a less redundant version of) it is added to the implication base. Otherwise, the partial context is extend with a counterexample from $\overline{\mathcal{K}}$ provided by the expert.

The following theorem states that this algorithm always terminates, and in which sense it is correct. In Section 4, we will briefly remark on its complexity.

Theorem 2.11 *Let M be a finite set of attributes, and $\overline{\mathcal{K}}$ and $\mathcal{K}_0$ respectively a full and a partial context over the attributes*

Algorithm 1 Attribute exploration for partial contexts

Input: $M = \{m_1, \ldots, m_n\}, \mathcal{K}_0$ {attribute set and partial context, realized by full context $\overline{\mathcal{K}}$}
$\mathcal{K} := \mathcal{K}_0$ {initialize partial context}
$\mathcal{L} := \varnothing$ {initial empty set of implications}
$P := \varnothing$ {lectically smallest $\mathcal{L}$-closed set}
while $P \neq M$ **do**
  Compute $\mathcal{K}(P)$
  if $P \neq \mathcal{K}(P)$ **then** $\{P \to \mathcal{K}(P)$ is undecided$\}$
    Ask the expert if $P \to \mathcal{K}(P)$ is refuted by $\overline{\mathcal{K}}$
    if no **then** $\{P \to \mathcal{K}(P)$ not refuted$\}$
      $\mathcal{K} := \mathcal{K}'$ where $\mathcal{K}'$ is a partial context such that $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$ {optionally extend $\mathcal{K}$}
      $\mathcal{L} := \mathcal{L} \cup \{P \to \mathcal{K}(P) \setminus P\}$
      $P_{\text{new}} := \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P <_j \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
    else $\{P \to \mathcal{K}(P)$ refuted$\}$
      Get a partial context $\mathcal{K}'$ from the expert such that $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$ and $P \to \mathcal{K}(P)$ is refuted by $\mathcal{K}'$
      $\mathcal{K} := \mathcal{K}'$
      $P_{\text{new}} := P$ {P not changed}
    end if
  else {trivial implication}
    $P_{\text{new}} := \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P <_j \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
  end if
  $P := P_{\text{new}}$
end while

in M such that $\mathcal{K}_0 \leq \overline{\mathcal{K}}$. Then Algorithm 1 terminates and, upon termination, it outputs a partial context $\mathcal{K}$ and a set of implications $\mathcal{L}$ such that

1. *$\mathcal{L}$ is a base for $Imp(\overline{\mathcal{K}})$, and*
2. *$\mathcal{K}$ refutes every implication that is refuted by $\overline{\mathcal{K}}$.*

3 Completing DL Knowledge Bases

In order to represent knowledge about an application domain using Description Logics (DLs) (see [Baader *et al.*, 2003] for more details and references), one usually first defines the relevant concepts of this domain, and then describes relationships between concepts and between individuals and concepts in the knowledge base. To construct concepts, one starts with a set N_C of *concept names* (unary predicates) and a set N_R of *role names* (binary predicates), and builds complex *concept descriptions* out of them by using the *concept constructors* provided by the particular *description language* being used. In addition, a set N_I of *individual names* is used to refer to domain elements. In this paper, we do not fix a specific set of constructors since our *results apply to arbitrary DLs* as long as they allow for the constructors conjunction and negation (see the upper part of Table 1). A *TBox* is a finite set of general concept inclusions (GCIs), and an *ABox* is a finite set of concept and role assertions (see the lower part of Table 1). A *knowledge base (KB)* consists of a TBox together with an ABox. The semantics of concept descriptions, TBoxes, and ABoxes is given in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, where $\Delta^{\mathcal{I}}$ (the *domain*) is a non-empty set, and $\cdot^{\mathcal{I}}$ (the *interpretation function*) maps each concept name $A \in N_C$ to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each role name $r \in N_R$ to a binary relation $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name $a \in N_I$ to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Concept descriptions C are also interpreted as sets $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, which are defined inductively, as seen in the semantics column of Table 1 for the constructors conjunction and negation. An interpretation $\mathcal{I}$ is a *model* of the TBox $\mathcal{T}$ (the ABox $\mathcal{A}$) if it satisfies all its GCIs (assertions) in the sense shown in the semantics column of the table. In case $\mathcal{I}$ is a model of both $\mathcal{T}$ and $\mathcal{A}$, it is also called a model of the knowledge base $(\mathcal{T}, \mathcal{A})$. If there is such a model, we call the KB consistent.

Name of constructor	Syntax	Semantics
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
general concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$r(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$

Table 1: Conjunction, negation, GCIs, and ABox assertions.

Given a KB $(\mathcal{T}, \mathcal{A})$, concept descriptions C, D, and an individual name a, the inference problems *subsumption* and *instance* are defined as follows: C is *subsumed* by D w.r.t. $\mathcal{T}$ ($C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$; and a is an *instance* of C w.r.t. $\mathcal{T}$ and $\mathcal{A}$ ($\mathcal{T}, \mathcal{A} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models of $(\mathcal{T}, \mathcal{A})$. For most DLs, these problems are decidable, and there exist highly optimized DL reasoners such as FaCT, RACER, and Pellet that can solve these problems for very expressive DLs on large practical KBs.

As mentioned above, our approach for completing DL knowledge bases applies to arbitary DLs, provided that the description language allows at least for conjunction and negation, the TBox formalism allows for GCIs, the ABox formalism allows for concept assertions, and the subsumption and the instance problem are decidable.

DLs and partial contexts

Let $(\mathcal{T}, \mathcal{A})$ be a consistent DL knowledge base, and M be a finite set of concept descriptions. An individual name a occurring in $\mathcal{A}$ gives rise to the *partial object description* $pod_{\mathcal{T},\mathcal{A}}(a, M) := (A, S)$ where

$$A := \{C \in M \mid \mathcal{T}, \mathcal{A} \models C(a)\} \quad \text{and}$$
$$S := \{C \in M \mid \mathcal{T}, \mathcal{A} \models \neg C(a)\},$$

and the whole ABox induces the partial context

$$\mathcal{K}_{\mathcal{T},\mathcal{A}}(M) := \{pod_{\mathcal{T},\mathcal{A}}(a, M) \mid a \text{ an individual name in } \mathcal{A}\}.$$

Note that $pod_{\mathcal{T},\mathcal{A}}(a, M)$ is indeed a pod since $(\mathcal{T}, \mathcal{A})$ was assumed to be consistent. Similarly, any element $d \in \Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ gives rise to the *full object description* $fod_{\mathcal{I}}(d, M) := (\overline{A}, \overline{S})$ where

$$\overline{A} := \{C \in M \mid d \in C^{\mathcal{I}}\}, \quad \overline{S} := \{C \in M \mid d \in (\neg C)^{\mathcal{I}}\},$$

and the whole interpretation induces the full context

$$\mathcal{K}_{\mathcal{I}}(M) := \{fod_{\mathcal{I}}(d, M) \mid d \in \Delta^{\mathcal{I}}\}.$$

Proposition 3.1 *Let $(\mathcal{T}, \mathcal{A}), (\mathcal{T}', \mathcal{A}')$ be DL KBs such that $\mathcal{T} \subseteq \mathcal{T}'$ and $\mathcal{A} \subseteq \mathcal{A}'$, M a set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}', \mathcal{A}')$. Then $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M) \leq \mathcal{K}_{\mathcal{T}',\mathcal{A}'}(M) \leq \mathcal{K}_{\mathcal{I}}(M)$.*

Next, we straightforwardly transfer the notion of refutation of an implication from partial (full) contexts to knowledge bases (interpretations).

Definition 3.2 *The implication $L \to R$ over the attributes M is* refuted *by the knowledge base $(\mathcal{T}, \mathcal{A})$ if it is refuted by $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$, and it is* refuted *by the interpretation $\mathcal{I}$ if it is refuted by $\mathcal{K}_{\mathcal{I}}(M)$. If an implication is not refuted by $\mathcal{I}$, then we say that it* holds in $\mathcal{I}$. *In addition, we say that $L \to R$* follows from $\mathcal{T}$ *if $\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$, where $\sqcap L$ and $\sqcap R$ respectively stand for the conjunctions $\bigsqcap_{C \in L} C$ and $\bigsqcap_{D \in R} D$.*

Obviously, the implication $L \to R$ holds in $\mathcal{I}$ iff $(\sqcap L)^{\mathcal{I}} \subseteq (\sqcap R)^{\mathcal{I}}$. As an immediate consequence of this fact, we obtain:

Proposition 3.3 *Let $\mathcal{T}$ be a TBox and $\mathcal{I}$ be a model of $\mathcal{T}$. If $L \to R$ follows from $\mathcal{T}$, then it holds in $\mathcal{I}$.*

Completion of DL KBs: formal definition and algorithm

We are now ready to define what we mean by a completion of a DL knowledge base. Intuitively, the knowledge base is supposed to describe an intended model. For a fixed set M of "interesting" concepts, the knowledge base is complete if it contains all the relevant knowledge about implications between these concepts. To be more precise, if an implication holds in the intended interpretation, then it should follow from the TBox, and if it does not hold in the intended interpretation, then the ABox should contain a counterexample. Based on the notions introduced above, this is formalized as follows.

Definition 3.4 *Let $(\mathcal{T}, \mathcal{A})$ be a consistent DL knowledge base, M a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}, \mathcal{A})$. Then $(\mathcal{T}, \mathcal{A})$ is* M-complete *(or* complete *if M is clear from the context)* w.r.t. *$\mathcal{I}$ if the following three statements are equivalent for all implications $L \to R$ over M:*

1. *$L \to R$ holds in $\mathcal{I}$;*
2. *$L \to R$ follows from $\mathcal{T}$;*
3. *$L \to R$ is not refuted by $(\mathcal{T}, \mathcal{A})$.*

Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a DL knowledge base and $\mathcal{I}$ a model of $(\mathcal{T}_0, \mathcal{A}_0)$. Then $(\mathcal{T}, \mathcal{A})$ is a completion *of $(\mathcal{T}_0, \mathcal{A}_0)$ if it is complete and extends $(\mathcal{T}_0, \mathcal{A}_0)$, i.e., $\mathcal{T}_0 \subseteq \mathcal{T}$ and $\mathcal{A}_0 \subseteq \mathcal{A}$.*

An adaptation of the attribute exploration algorithm for partial contexts presented above can be used to compute a completion of a given knowledge base $(\mathcal{T}_0, \mathcal{A}_0)$ w.r.t. a fixed model $\mathcal{I}$ of this knowledge base. It is assumed that the *expert* has or can obtain enough information about this model to be able to answer questions of the form "Is $L \to R$ refuted by $\mathcal{I}$?". If the answer is "no," then $L \to R$ holds according to the expert's opinion, and is thus added to the implication base computed by the algorithm. In addition, the GCI $\sqcap L \sqsubseteq \sqcap R$ is added to the TBox. Since $L \to R$ is not refuted by $\mathcal{I}$, the interpretation $\mathcal{I}$ is still a model of the new TBox obtained this way. If the answer is "yes," then the expert is asked to extend the current ABox (by adding appropriate assertions on either old or new individual names) such that the extended ABox refutes $L \to R$ and $\mathcal{I}$ is still a model of this ABox. Because of Proposition 3.3, before actually asking the expert whether the implication $L \to R$ is refuted by $\mathcal{I}$, we can first check whether $\sqcap L \sqsubseteq \sqcap R$ already follows from the current TBox. If this is the case, then we know that $L \to R$ cannot be refuted by $\mathcal{I}$. This completion algorithm for DL knowledge bases is described in more detail in Algorithm 2.

Algorithm 2 Completion of DL knowledge bases

Input: $M = \{m_1, \ldots, m_n\}$, $(\mathcal{T}_0, \mathcal{A}_0)$ {attribute set and knowledge base, with model $\mathcal{I}$}
$\mathcal{T} := \mathcal{T}_0$, $\mathcal{A} := \mathcal{A}_0$
$\mathcal{L} := \varnothing$ {initial empty set of implications}
$P := \varnothing$ {lectically smallest $\mathcal{L}$-closed subset of M}
while $P \neq M$ **do**
  Compute $\mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$
  if $P \neq \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ **then** {check whether the implication follows from $\mathcal{T}$}
    if $\sqcap P \sqsubseteq_{\mathcal{T}} \sqcap \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ **then**
      $\mathcal{L} := \mathcal{L} \cup \{P \to \mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P\}$
      $P_{\text{new}} := \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
        for the max. j that satisfies
        $P <_j \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
    else
      Ask expert if $P \to \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is refuted by $\mathcal{I}$.
      if no **then** {$\sqcap P \sqsubseteq \sqcap \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is satisfied in $\mathcal{I}$}
        $\mathcal{L} := \mathcal{L} \cup \{P \to \mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P\}$
        $P_{\text{new}} := \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
          for the max. j that satisfies
          $P <_j \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
        $\mathcal{T} := \mathcal{T} \cup \{\sqcap P \sqsubseteq \sqcap(\mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P)\}$
      else
        Get an ABox $\mathcal{A}'$ from the expert such that
          $\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{I}$ is a model of $\mathcal{A}'$, and
          $P \to \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is refuted by $\mathcal{A}'$
        $\mathcal{A} := \mathcal{A}'$ {extend the ABox}
        $P_{\text{new}} := P$ {P not changed}
      end if
    end if
  else
    $P_{\text{new}} := \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
      for the max. j that satisfies
      $P <_j \mathcal{L}((P \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
  end if
  $P := P_{\text{new}}$
end while

Note that Algorithm 2 applied to $\mathcal{T}_0, \mathcal{A}_0, M$ with the underlying model $\mathcal{I}$ of $(\mathcal{T}_0, \mathcal{A}_0)$ is an instance of Algorithm 1 applied to the partial context $\mathcal{K}_{\mathcal{T}_0,\mathcal{A}_0}(M)$ with the underlying full context $\mathcal{K}_{\mathcal{I}}(M)$ as realizer. This shows that Theorem 2.11 applies also to Algorithm 2, which implies:

Theorem 3.5 *Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a consistent knowledge base, M a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}_0, \mathcal{A}_0)$, and let $(\mathcal{T}, \mathcal{A})$ be the knowledge base computed by*

Algorithm 2. Then $(\mathcal{T}, \mathcal{A})$ *is a completion of* $(\mathcal{T}_0, \mathcal{A}_0)$.

4 Conclusion

We have extended the attribute exploration algorithm from FCA to partial contexts, and have shown how the extended algorithm can be used to complete DL knowledge bases, using both DL reasoning and answers given by a domain expert. This algorithm inherits its complexity from "classical" attribute exploration [Ganter and Wille, 1999]: in the worst case, which occurs if there are few or many relationships between attributes, it is exponential in the number of attributes. Regarding the number of questions asked to the expert, it easily follows from results shown in the technical report that our method asks the minimum number of questions with positive answers. For the questions with negative answers, the behaviour depends on the answers given by the expert: FCA-theory implies that there always exist counterexamples that, if taken in each step, ensure a minimal number of questions with negative answers. In general, however, one cannot assume that the expert provides these "best" counterexamples.

Based on the results presented in the previous two sections, we have implemented a first experimential version of a tool for completing DL knowledge bases as an extension of the ontology editor Swoop [Kalyanpur *et al.*, 2006a], which uses the system Pellet as underlying reasoner [Sirin and Parsia, 2004]. We have just started to evaluate our tool on the OWL ontology for human protein phosphatases mentioned in the introduction, with biologists as experts, and hope to get first significant results on its usefulness and performance in the near future. Unsurprisingly, we have observed that the experts sometimes make errors when answering implication questions. Hence we will extend the completion tool such that it supports detecting such errors and also allows to correct errors without having to restart the exploration from scratch.

From a theoretical point of view, we will also look at extensions of our definition of a complete KB. As a formalization of what "all relationships between interesting concepts" really means, we have used subsumption relationships between conjunctions of elements of the set of interesting concepts M. One could also consider more complex relationships by fixing a specific DL $\mathcal{D}$, and then taking, as attributes, all $\mathcal{D}$-concept descriptions over the concept "names" from M. The immediate disadvantage of this extension is that, in general, the set of attributes becomes infinite, and thus termination of the exploration process is no longer guaranteed. An extension of classical attribute exploration (i.e., for full contexts) in this direction is described in [Rudolph, 2004]. The main idea to deal with the problem of an infinite attribute set used there is to restrict the attention to concept descriptions with a bounded role depth. But even though this makes the attribute set finite, its size is usually too large for practical purposes. Thus, an adaptation of Rudolph's method to our purposes not only requires extending this method to partial contexts, but also some new approach to reduce the number of attributes.

References

[Baader *et al.*, 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Baader *et al.*, 2006] F. Baader, B. Ganter, U. Sattler, and B. Sertkaya. Completing description logic knowledge bases using formal concept analysis. LTCS Report 06-02, Theoretical Computer Science, TU Dresden, 2006. Available at http://lat.inf.tu-dresden.de/research/reports.html

[Burmeister and Holzer, 2005] P. Burmeister and R. Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis*, volume 3626 of *LNCS*. Springer, 2005.

[Ganter and Wille, 1999] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1999.

[Horrocks *et al.*, 2003] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *J. of Web Semantics*, 1(1), 2003.

[Kalyanpur *et al.*, 2006a] A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. Hendler. Swoop: A Web ontology editing browser. *J. of Web Semantics*, 4(2), 2006.

[Kalyanpur *et al.*, 2006b] A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in OWL ontologies. In *Proc. of ESWC'06*, volume 4011 of *LNCS*. Springer, 2006.

[Knublauch *et al.*, 2004] H. Knublauch, R. W. Fergerson, N. F. Noy, and M. A. Musen. The Protégé OWL plugin: An open development environment for semantic web applications. In *Proc. of ISWC 2004*, volume 3298 of *LNCS*. Springer, 2004.

[Oberle *et al.*, 2004] D. Oberle, R. Volz, B. Motik, and S. Staab. An extensible ontology software environment. *Handbook on Ontologies*, International Handbooks on Information Systems. Springer, 2004.

[Obiedkov, 2002] S. A. Obiedkov. Modal logic for evaluating formulas in incomplete contexts. In *Proc. of ICCS 2002*, volume 2393 of *LNCS*. Springer, 2002.

[Rudolph, 2004] S. Rudolph. Exploring relational structures via $\mathcal{FLE}$. In *Proc. of ICCS'04*, volume 3127 of *LNCS*. Springer, 2004.

[Schlobach and Cornet, 2003] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI 2003*. Morgan Kaufmann, 2003.

[Sirin and Parsia, 2004] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of DL'04*, 2004.

[Wille, 1982] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In *Ordered Sets*. Reidel, Dordrecht-Boston, 1982.

[Wolstencroft *et al.*, 2005] K. Wolstencroft, A. Brass, I. Horrocks, P. W. Lord, U. Sattler, D. Turi, and R. Stevens. A little semantic web goes a long way in biology. In *Proc. of ISWC'05*, volume 3729 of *LNCS*. Springer, 2005.