# Usability Issues in Description Logic Knowledge Base Completion

Franz Baader and Barış Sertkaya*

TU Dresden, Germany
{baader,sertkaya}@tcs.inf.tu-dresden.de

**Abstract.** In a previous paper, we have introduced an approach for extending both the terminological and the assertional part of a Description Logic knowledge base by using information provided by the assertional part and by a domain expert. This approach, called knowledge base completion, was based on an extension of attribute exploration to the case of partial contexts. The present paper recalls this approach, and then addresses usability issues that came up during first experiments with a preliminary implementation of the completion algorithm. It turns out that these issues can be addressed by extending the exploration algorithm for partial contexts such that it can deal with implicational background knowledge.

## 1 Introduction

Description Logics (DLs) [1] 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 [20]. As a consequence of this standardization, several ontology editors support OWL [19,22,23,27], 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 [6,7,21,24,32]. 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). Here, we are concerned with a different quality dimension, namely *completeness* of the ontology w.r.t. to the intended application domain. In [5],

* Supported by the German Research Foundation (DFG) under grant BA 1122/12-1.

© Springer-Verlag Berlin Heidelberg 2009

we have provided 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 in extending the ontology appropriately if this is not the case. In the present paper, we give an overview over this approach, and then describe how the general framework must be extended such that it becomes easier to use for a domain expert. But first, let us introduce the problem of knowledge base completion, and our approach for solving it, in a bit more detail.

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 [37]. This ontology was developed based on information from peer-reviewed publications. The human protein phosphatase family has been well characterized 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 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

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

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 [11] is an approach developed in Formal Concept Analysis (FCA) [14] 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 [29,36], 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 could not directly use this approach in [5]. The main reason for this 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. In contrast, in a *partial context*, a property may be known to hold or not to hold for an individual, but there is also the third possibility that nothing is known about the relationship between the individual and this property.

There has been some work on how to extend FCA and attribute exploration from complete knowledge to the case of such partial knowledge [9,10,17,18,28]. 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. From the FCA point of view, the main new result in [5] is the extension of attribute exploration to the case of partial contexts. This approach is then used to derive an algorithm for completing DL knowledge bases.

Before publishing [5], we had implemented a first experimental version of a tool for completing DL knowledge bases as an extension of the ontology editor Swoop [22], using the system Pellet as underlying reasoner [33]. A first evaluation of this tool on the OWL ontology for human protein phosphatases mentioned in the introduction, with biologists as experts, was quite promising, but also showed that the tool must be improved in order to be useful in practice. In particular, we have observed that the experts sometimes make errors when answering queries. Thus, the tool should support the expert in detecting such errors, and also make it possible to correct errors without having to restart the exploration process from scratch. Another usability issue on the wish list of our experts was to allow the postponement of answering certain queries, while continuing the exploration process with other queries.

The present paper is a follow-up work to [5], which addresses these usability issues. In the next section, we give a brief introduction to DLs, and then recall, in Section 3, the completion approach developed in [5]. For more details, we refer the reader to that paper as well as to the technical report [4] accompanying it. In Section 4, we address the usability issues mentioned above. From the FCA point of view, our solution to these problems requires an extension of the results in [5] to the case of attribute exploration *w.r.t. background knowledge and partial contexts*. In Section 5, we describe our implementation of the improved approach, which is now realized as an OWL plugin to PROTÉGÉ 4 [19], and uses the incremental reasoning facilities of Pellet [16].

## 2 Description Logics

In order to represent knowledge about an application domain using Description Logics (DLs) one usually first defines the relevant concepts of this domain, and then describes relationships between concepts and relationships 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. Table 1 displays commonly used concept constructors. In this table $C$ and $D$ stand for concept descriptions, $r$ for a role name, and $a, b, a_1, \ldots, a_n$ for individual names. In the current 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). For our purposes, 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. As usual, we use $C \equiv D$ as an abbreviation for the two GCIs $C \sqsubseteq D$ and $D \sqsubseteq C$.

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

**Table 1.** Syntax and semantics of commonly used constructors

| Constructor name | Syntax | Semantics |
|---|---|---|
| negation | $\neg C$ | $\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$ |
| conjunction | $C \sqcap D$ | $C^{\mathcal{I}} \cap D^{\mathcal{I}}$ |
| disjunction | $C \sqcup D$ | $C^{\mathcal{I}} \cup D^{\mathcal{I}}$ |
| existential restriction | $\exists r.C$ | $\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$ |
| one-of | $\{a_1, \ldots, a_n\}$ | $\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$ |
| 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 | $r(a,b)$ | $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$ |

$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, 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. 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 called a model of the knowledge base $(\mathcal{T}, \mathcal{A})$.

Given a KB $(\mathcal{T}, \mathcal{A})$, concept descriptions $C, D$, and an individual name $a$, the inference problems *subsumption*, *instance*, and *consistency* are defined as follows:

- *Subsumption:* $C$ is *subsumed* by $D$ w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$
- *Instance:* $a$ is an *instance* of $C$ w.r.t. $\mathcal{T}$ and $\mathcal{A}$ (written $\mathcal{T}, \mathcal{A} \models C(a)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ holds for all models of $(\mathcal{T}, \mathcal{A})$.
- *Consistency:* the knowledge base $(\mathcal{T}, \mathcal{A})$ is *consistent* if it has a model.

For most DLs, these problems are decidable, and there exist highly optimized DL reasoners such as FaCT++ [35], RacerPro [15], Pellet [33], KAON2 [25], and HermiT [26], which can solve these problems for very expressive DLs on large knowledge bases from practical applications.

The following example demonstrates how a DL that has the constructors conjunction, disjunction, existential restriction, and one-of can be used to model some simple properties of countries.

*Example 1.* Assume that our set of concept names $N_C$ contains the concepts Country, Ocean, and Sea; the set of role names $N_R$ contains the roles hasBorderTo, isMemberOf, and hasOfficialLanguage; and the set of individual names $N_I$ contains the individuals German, EU, MediterraneanSea, Italy, and Germany. Using these names, the following TBox defines a coastal country as a country that has a border to a sea or an ocean; a Mediterranean country as a country that has border to the MediterraneanSea; a German-speaking country as a country that has the language German as an official language; and an EU member as a country that is a member of the EU.

$$\begin{aligned}\mathcal{T}_{countries} := \{ \ \mathsf{Coastal} &\equiv \mathsf{Country} \sqcap \exists \mathsf{hasBorderTo}.(\mathsf{Ocean} \sqcup \mathsf{Sea})\\ \mathsf{EUmember} &\equiv \mathsf{Country} \sqcap \exists \mathsf{isMemberOf}.\{\mathsf{EU}\}\\ \mathsf{Mediterranean} &\equiv \mathsf{Country} \sqcap \exists \mathsf{hasBorderTo}.\{\mathsf{MediterraneanSea}\}\\ \mathsf{GermanSpeaking} &\equiv \mathsf{Country} \sqcap \exists \mathsf{hasOfficialLanguage}.\{\mathsf{German}\} \ \}\end{aligned}$$

The following ABox states facts about the countries Italy and Germany, and about the Mediterranean Sea:

$$\begin{aligned}\mathcal{A}_{countries} := \{&\mathsf{GermanSpeaking}(\mathsf{Germany}),\ \mathsf{EUmember}(\mathsf{Germany}),\\ &\mathsf{Coastal}(\mathsf{Germany}),\ \mathsf{Mediterranean}(\mathsf{Italy}),\ \mathsf{Sea}(\mathsf{MediterraneanSea})\}\end{aligned}$$

## 3 Partial Contexts, Attribute Exploration, and Completion of DL Knowledge Bases

In [5], we have extended the classical approach to FCA to the case of objects that have only a partial description in the sense that, for some attributes, it is not known whether they are satisfied by the object or not. This was needed due to the open-world semantics of DL knowledge bases. If an assertion $C(a)$ does not follow from a knowledge base, then one does not assume that its negation holds. Thus, if neither $C(a)$ nor $\neg C(a)$ follows, then we do not know whether $a$ has the property $C$ or not. In this section, we first give the basic definitions for extending FCA to the case of partially described objects, and then introduce a version of attribute exploration that works in this setting. More details can be found in [5,4]. In the following, we assume that we have a finite set $M$ of attributes and a (possibly infinite) set of objects.

**Definition 1.** *A* partial object description (pod) *is a tuple* $(A, S)$ *where* $A, S \subseteq M$ *are such that* $A \cap S = \emptyset$. *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.** *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 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 $R \cap S \neq \emptyset$. 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 $Mod(\mathcal{L})$.*

Obviously, $\mathcal{K} \leq \mathcal{K}'$ implies that every implication refuted by $\mathcal{K}$ is also refuted by $\mathcal{K}'$. 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
- $L \rightarrow R \in \mathcal{L}$ and $L \subseteq Q$ imply $R \subseteq Q$.

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

**Definition 4.** *The implication $L \rightarrow R$ 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 fact is trivial, but turns out to be crucial for our attribute exploration algorithm.

**Proposition 1.** *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 $P \rightarrow \mathcal{K}(P)$ is not refuted by $\mathcal{K}$.*

## Attribute Exploration with Partial Contexts

The classical attribute exploration algorithm of FCA [11,14] 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 [9,17,18,10], 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 "$L \rightarrow R$?" w.r.t. the full context $\overline{\mathcal{K}}$. More precisely, if the answer is "yes," then $\overline{\mathcal{K}}$ does not refute $L \rightarrow R$. The implication $L \rightarrow R$ is then added to $\mathcal{L}$. Otherwise, the expert extends the current context $\mathcal{K}$ such that the extended context refutes $L \rightarrow R$ and still has $\overline{\mathcal{K}}$ as a realizer. Consequently, the following invariant will be satisfied by $\mathcal{K}, \overline{\mathcal{K}}, \mathcal{L}$: $\quad \mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$.

Since $\overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ implies $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$, this invariant ensures that $\mathcal{L}$ is sound for $Imp(\overline{\mathcal{K}})$. Our aim is to enrich $\mathcal{K}$ and $\mathcal{L}$ such that eventually $\mathcal{L}$ is also complete for $Imp(\overline{\mathcal{K}})$, 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 5.** *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 implication 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 [4].

**Proposition 2.** *Assume that $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ and that all implications are decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. Then $\mathcal{L}$ is complete for $Imp(\overline{\mathcal{K}})$ and $\mathcal{K}$ refutes all implications not belonging to $Imp(\overline{\mathcal{K}})$.*

How can we find—and let the expert decide—all undecided implications without naively considering all implications? The following proposition motivates why it is sufficient to consider implications whose left-hand sides are $\mathcal{L}$-closed. It is an immediate consequence of the fact that $\mathcal{L}(\cdot)$ is a closure operator, and thus idempotent.

**Proposition 3.** *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 1 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 [14]. In this approach, the elements of $M$ are assumed to have a fixed order that imposes a linear order on the power set of $M$, called the *lectic order*. An algorithm that, given $\mathcal{L}$ and an $\mathcal{L}$-closed set $P$, computes in polynomial time the lectically next $\mathcal{L}$-closed set that comes after $P$, is described in [11].

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, $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$ 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, an attribute exploration algorithm for partial contexts was introduced in [5], and is here recalled as Algorithm 1.

The following theorem states that this algorithm always terminates, and in which sense it is correct.

**Theorem 1.** *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 in $M$ such that $\mathcal{K}_0 \leq \overline{\mathcal{K}}$. Then*

**Algorithm 1.** Attribute exploration for partial contexts

```
1: Input: $M = \{m_1, \ldots, m_n\}, \mathcal{K}_0$    {attribute set and partial context, realized by full context $\overline{\mathcal{K}}$.}
2: $\mathcal{K} := \mathcal{K}_0$    {initialize partial context.}
3: $\mathcal{L} := \emptyset$    {initial empty set of implications.}
4: $P := \emptyset$    {lectically smallest $\mathcal{L}$-closed set.}
5: while $P \neq M$ do
6:   Compute $\mathcal{K}(P)$
7:   if $P \neq \mathcal{K}(P)$ then {$P \to \mathcal{K}(P)$ is undecided.}
8:     Ask the expert if $P \to \mathcal{K}(P)$ is refuted by $\overline{\mathcal{K}}$
9:     if no then {$P \to \mathcal{K}(P)$ not refuted.}
10:      $\mathcal{K} := \mathcal{K}'$ where $\mathcal{K}'$ is a partial context such that
         $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$    {optionally extend $\mathcal{K}$.}
11:      $\mathcal{L} := \mathcal{L} \cup \{P \to \mathcal{K}(P) \setminus P\}$
12:      $P_{\text{new}}$ := lectically next $\mathcal{L}$-closed set after $P$
13:    else {$P \to \mathcal{K}(P)$ refuted.}
14:      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 $\mathcal{K}'$
15:      $\mathcal{K} := \mathcal{K}'$
16:      $P_{\text{new}} := P$    {$P$ not changed.}
17:    end if
18:  else {trivial implication.}
19:    $P_{\text{new}}$ := lectically next $\mathcal{L}$-closed set after $P$
20:  end if
21:  $P := P_{\text{new}}$
22: end while
```

*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 $\overline{\mathcal{K}}$.*

## 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

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

The whole ABox induces the partial context

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

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

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

The whole interpretation induces the full context

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

**Proposition 4.** *Let* $(\mathcal{T}, \mathcal{A}), (\mathcal{T}', \mathcal{A}')$ *be DL knowledge bases 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* $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M) \leq \mathcal{K}_{\mathcal{T}',\mathcal{A}'}(M) \leq \mathcal{K}_{\mathcal{I}}(M)$*.*

We can straightforwardly transfer the notion of refutation of an implication from partial (full) contexts to knowledge bases (interpretations).

**Definition 6.** *The implication* $L \rightarrow R$ *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* $L \rightarrow R$ 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 $L \rightarrow R$ holds in $\mathcal{I}$ iff $(\sqcap L)^{\mathcal{I}} \subseteq (\sqcap R)^{\mathcal{I}}$. As an immediate consequence of this fact, we obtain:

**Proposition 5.** *Let* $\mathcal{T}$ *be a TBox and* $\mathcal{I}$ *be a model of* $\mathcal{T}$*. If* $L \rightarrow R$ *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. Based on the notions introduced above, this is formalized as follows.

**Definition 7.** *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* $L \rightarrow R$ *over* $M$*:*

1. $L \rightarrow R$ *holds in* $\mathcal{I}$*;*
2. $L \rightarrow R$ *follows from* $\mathcal{T}$*;*
3. $L \rightarrow R$ *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 an* M-completion of $(\mathcal{T}_0, \mathcal{A}_0)$ w.r.t. $\mathcal{I}$ *if it is M-complete w.r.t.* $\mathcal{I}$ *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 $L \rightarrow R$ refuted by $\mathcal{I}$?". If the answer is "no," then $L \rightarrow R$ 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 $L \rightarrow R$ 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 $L \rightarrow R$ and $\mathcal{I}$ is still a model of this ABox.

It is possible to optimize this algorithm by employing DL reasoning. Because of Proposition 5, before actually asking the expert whether the implication $L \rightarrow R$ 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 $L \rightarrow R$ cannot be refuted by $\mathcal{I}$, and we tacitly accept and add this implication to the current set of implications $\mathcal{L}$. Similarly, there are also cases where an implication can be rejected without asking the expert because accepting it would make the knowledge base inconsistent. However, in this case the expert still needs to extend the ABox such that the implication is refuted. The following example illustrates this case, which was not taken into account in the original version of the completion algorithm in [5].

*Example 2.* Consider the knowledge base $(\mathcal{T}, \mathcal{A})$ with the empty TBox $\mathcal{T} = \emptyset$, and the ABox $\mathcal{A} = \{(\exists r.A \sqcap \forall r.\neg B)(a)\}$. Clearly, the implication $\{A\} \rightarrow \{B\}$ does not follow from $\mathcal{T}$. Moreover, it is not refuted by $\mathcal{A}$ because this ABox does not explicitly contain a named individual that is an instance of both $A$ and $\neg B$. That is, the implication $\{A\} \rightarrow \{B\}$ is undecided. However, if the user accepted this implication, and thus the GCI $A \sqsubseteq B$ were added to $\mathcal{T}$, the knowledge base would become inconsistent since the assertion in $\mathcal{A}$ enforces the existence of an implicit individual that belongs to both $A$ and $\neg B$. This shows that this GCI cannot hold in the underlying model $\mathcal{I}$ of $(\mathcal{T}, \mathcal{A})$, and thus it is refuted in the full context $\mathcal{K}_{\mathcal{I}}(M)$.

The improved completion algorithm for DL knowledge bases obtained from these considerations is described in Algorithm 2. 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. For this reason, the next theorem is an easy consequence of Theorem 1.

**Theorem 2.** *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 $(\mathcal{T}_0, \mathcal{A}_0)$.*

Let us demonstrate the execution of Algorithm 2 on an extension of the knowledge base constructed in Example 1, where $M$ consists of the concepts $\mathsf{Coastal}$, $\mathsf{Mediterranean}$, $\mathsf{EUmember}$, and $\mathsf{GermanSpeaking}$, the ABox contains additional information on some countries, and $\mathcal{I}$ is the "real world."

*Example 3.* Let the partial context derived from the initial ABox be the one depicted in Table 2. Given this ABox, and the TBox in Example 1, the first implication question posed to the expert is $\{\mathsf{GermanSpeaking}\} \rightarrow \{\mathsf{EUmember}, \mathsf{Coastal}\}$.

**Algorithm 2.** Completion of DL knowledge bases

```
1:  Input: $M = \{m_1, \ldots, m_n\}$, $(\mathcal{T}_0, \mathcal{A}_0)$    {attribute set; KB with model $\mathcal{I}$.}
2:  $\mathcal{T} := \mathcal{T}_0$,   $\mathcal{A} := \mathcal{A}_0$
3:  $\mathcal{L} := \emptyset$    {initial empty set of implications.}
4:  $P := \emptyset$    {lectically smallest $\mathcal{L}$-closed subset of $M$.}
5:  while $P \neq M$ do
6:    Compute $\mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$
7:    if $P \neq \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ then {check whether the implication follows from $\mathcal{T}$.}
8:      if $\sqcap P \sqsubseteq_{\mathcal{T}} \sqcap \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ then
9:        $\mathcal{L} := \mathcal{L} \cup \{P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P\}$
10:       $P_{\text{new}}$ := lectically next $\mathcal{L}$-closed set after $P$
11:     else
12:       if $(\mathcal{T} \cup \{\sqcap P \sqsubseteq \sqcap \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)\}, \mathcal{A})$ is inconsistent then
13:         Get an ABox $\mathcal{A}'$ from the expert such that $\mathcal{A} \subseteq \mathcal{A}'$, $\mathcal{I}$ is a model of $\mathcal{A}'$,
              and $P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is refuted by $\mathcal{A}'$
14:         $\mathcal{A} := \mathcal{A}'$    {extend the ABox.}
15:       else
16:         Ask expert if $P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is refuted by $\mathcal{I}$.
17:         if no then {$\sqcap P \sqsubseteq \sqcap \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is satisfied in $\mathcal{I}$.}
18:           $\mathcal{L} := \mathcal{L} \cup \{P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P\}$
19:           $P_{\text{new}}$ := lectically next $\mathcal{L}$-closed set after $P$
20:           $\mathcal{T} := \mathcal{T} \cup \{\sqcap P \sqsubseteq \sqcap(\mathcal{K}_{\mathcal{T},\mathcal{A}}(P) \setminus P)\}$
21:         else
22:           Get an ABox $\mathcal{A}'$ from the expert such that $\mathcal{A} \subseteq \mathcal{A}'$,
                $\mathcal{I}$ is a model of $\mathcal{A}'$, and $P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is refuted by $\mathcal{A}'$
23:           $\mathcal{A} := \mathcal{A}'$    {extend the ABox.}
24:         end if
25:       end if
26:     end if
27:   else {trivial implication.}
28:     $P_{\text{new}}$ := lectically next $\mathcal{L}$-closed set after $P$
29:   end if
30:   $P := P_{\text{new}}$
31: end while
```

The answer is "no," since Austria is German-speaking, but it is not a coastal country. Assume that the expert turns Austria into a counterexample by asserting that it is German-speaking. The second question is then whether the implication {GermanSpeaking} → {EUmember} holds. The answer is again "no" since Switzerland is a German-speaking country, but not an EU member. Assume that the expert adds the new individual Switzerland to the ABox, and asserts that it is an instance of GermanSpeaking and ¬EUmember. The next question is {Mediterranean} → {EUmember, Coastal}. The answer is again "no" because Turkey is a Mediterranean country, but it is not an EU member. Assume that the expert adds the individual Turkey to the ABox, and asserts that it is an instance of ¬EUmember. The next question {Mediterranean} → {Coastal} follows from the TBox. Thus, it is not posed to the expert, and the algorithm continues

**Table 2.** The partial context before completion

| | Coastal | Mediterranean | EUmember | GermanSpeaking |
|---|---|---|---|---|
| Italy | + | + | + | − |
| India | + | − | − | − |
| Germany | + | − | + | + |
| Austria | − | − | + | ? |

**Table 3.** The partial context after completion

| | Coastal | Mediterranean | EUmember | GermanSpeaking |
|---|---|---|---|---|
| Italy | + | + | + | − |
| India | + | − | − | − |
| Germany | + | − | + | + |
| Austria | − | − | + | + |
| Switzerland | − | − | − | + |
| Turkey | + | + | − | − |

with the last question $\{\mathsf{Coastal}, \mathsf{GermanSpeaking}\} \rightarrow \{\mathsf{EUmember}\}$. The answer to this question is "yes" because the only countries that are both coastal and German-speaking (Germany and Belgium) are also EU members. Thus the GCI $\mathsf{Coastal} \sqcap \mathsf{GermanSpeaking} \sqsubseteq \mathsf{EUmember}$ is added to the TBox, and the completion process is finished. The completion yields the final context displayed in Table 3.

## 4 Improving the Usability of the Completion Procedure

Based on on the approach described in the previous section, we had implemented a first experimental version of a DL knowledge base completion tool. Our experiments with this tool showed that during completion the expert sometimes makes errors. For better usability of the completion procedure, it is important to support the expert in detecting and correcting these errors. Moreover, although we assume that the expert is omniscient (i.e., is potentially able to answer all implication questions), we have observed that it is convenient to be able to defer a question and answer it later. In the following, we discuss these problems in more detail and show how we address them in our improved completion tool.

### Detecting Errors

We say that the expert makes an *error* if he extends the knowledge base such that it no longer has the underlying model $\mathcal{I}$ as its model. Since the procedure has no direct access to $\mathcal{I}$, in general it cannot detect such errors without help from the expert. The only case were the procedure can automatically detect that an error has occurred is when the knowledge base becomes inconsistent. Obviously, the underlying model $\mathcal{I}$ cannot be a model of an inconsistent KB.

However, when an inconsistency is detected by DL reasoning, then it is not clear at which stage the actual error was made. In fact, although only the last extension of the knowledge base has made it inconsistent, the deviation from what holds in $\mathcal{I}$ may have occurred in a previous step. Pinpointing [6,7,21,24,32] can be used to compute all minimal subsets of the knowledge base that are already inconsistent, and thus help the expert to find the place where the error was made. But in the end, the expert needs to tell the completion tool which are the erroneous assertions and/or GCIs.

The expert may also be alerted by DL reasoning to errors in cases where the knowledge base is not inconsistent. In fact, after each extension of the KB, the DL reasoner re-classifies it, i.e., computes the implied subsumption and instance relationships. If one of them contradicts the experts knowledge about $\mathcal{I}$, she also knows that an error has occurred. Again, pinpointing can show all minimal subsets of the knowledge base from which this unintended consequence already follows.

## Recovery from Errors

Once the sources of the error are found, the next task is to correct it without producing unnecessary extra work for the expert. Of course, one can just go back to the step where the first error was made, and continue the completion process from there, this time with a correct answer. The problem with this simple approach is that it throws away all the information about $\mathcal{I}$ that the expert has added (by answering implication queries) after the first error had occurred. Consequently, the completion procedure may again pose implication queries whose answer actually follows from this information. On the DL side, it is no problem to keep those assertions and GCIs that really hold in $\mathcal{I}$. More precisely, the completion procedure can keep the GCIs and assertions for which the expert has stated that they are not erroneous. In fact, our completion procedure allows for arbitrary extensions of the KB as long as the KB stays a model of $\mathcal{I}$.

On the FCA side, it is less clear whether one can keep the implications that have been added after the first error had been made. In fact, since the new (correct) answer differs from the previous incorrect one, the completion procedure may actually produce different implication questions. The proof of correctness of the procedure, as given in [4], strongly depends on the fact that implications are enumerated according to the lectic ordering of their left-hand sides. Thus, having implications in the implication set for which the left-hand side is actually larger than the left-hand side of the implication currently under consideration may potentially cause problems. However, not using these implications may lead to more implication questions being generated. Fortunately, this problem can be solved by using these implications as background knowledge [34] rather than as part of the implication base to be generated. Thus, when correcting an error, we move implications generated after the error had occurred, but marked by the expert as correct, to the background knowledge. Correctness then follows from the fact that Stumme's extension of attribute exploration to the case of implicational background knowledge [34] can further be extended to partial contexts (see the corresponding subsection below).

**Deferring Questions**

Although we assume that the expert is omniscient in the sense that she is potentially able to answer all implication questions, it is convenient to be able to defer answering certain questions. For example, in the biology application mentioned in the introduction, answering an implication question may necessitate searching the literature on human protein phosphatases, querying gene and protein databases, or even making new experiments. This research may take quite some time, and can possibly be delegated to other researchers. It would thus be good if the expert could in parallel continue with the completion process.

Our approach for achieving this is that we allow the expert to stop the completion process and change the linear order on the set $M$ of interesting concepts. This results in a different lectic order, and thus other implication questions may be asked before the problematic one.[2] To ensure that correctness is not compromised by this approach, we leave the knowledge base as it is, but move all the implications collected so far to the background knowledge. The completion procedure is then restarted with the first set that is closed w.r.t. the background implications, i.e., the closure of $\emptyset$.

**Attribute Exploration with Background Knowledge for Partial Contexts**

Our approaches for recovering from errors and for allowing the expert to defer answering a question depend on the use of implications as background knowledge. Attribute exploration in the presence of implicational background knowledge [34] and also non-implicational background knowledge [12,13] has already been considered in the literature for full contexts. For partial context, it has been considered in [17]. However, as mentioned before, this work is based on assumptions that are different from ours, and thus cannot directly be used.

In [4] we have shown termination and correctness of Algorithm 1 under the condition that the initial set of implications $\mathcal{L}_0$ is empty (line 3). In the following we show that Algorithm 1 stays correct and terminating if we modify it such that

1. the initial set of implications $\mathcal{L}_0$ already contains background implications that are not refuted by $\overline{\mathcal{K}}$, i.e., satisfies $\overline{\mathcal{K}} \subseteq \mathit{Mod}(\mathcal{L}_0)$ (line 3); and
2. the variable $P$ is initialized with the lectically smallest $\mathcal{L}_0$-closed set, i.e., with $\mathcal{L}_0(\emptyset)$ rather than with $\emptyset$ (line 4).

**Theorem 3.** *Let $M$ be a finite set of attributes, $\overline{\mathcal{K}}$ and $\mathcal{K}_0$ respectively a full and a partial context over the attributes in $M$ such that $\mathcal{K}_0 \leq \overline{\mathcal{K}}$, $\mathcal{L}_0$ an initial set of background implications such that $\overline{\mathcal{K}} \subseteq \mathit{Mod}(\mathcal{L}_0)$, and $P_0$ the lectically smallest $\mathcal{L}_0$-closed set $\mathcal{L}_0(\emptyset)$. Then the modified Algorithm 1 terminates on this input and upon termination it outputs a partial context $\mathcal{K}$ and a set of implications $\mathcal{L}$ such that*

[2] Note, however, that this need not always be the case, i.e., it could be that also with the changed order the problematic question is the next one.

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

Since the proof of this theorem is almost identical to the one of Theorem 1, we only give a sketch (see the proof of Theorem 1 given in [4] for details).

*Termination* is based on the following observation: in every iteration of the algorithm, one

(a) either considers as left-hand side of the current implication a new set $P$ that is lectically larger than the previous one,
(b) or stays with the same left-hand side $P$, but decreases the cardinality of the right-hand side.

Thus, both iterations of the form (a) and (b) cannot occur infinitely often. This argument applies unchanged to the modified algorithm.

To show *correctness*, we must show 1. and 2. in the statement of the theorem. Thus, we must show that $\mathcal{L}$ is both sound and complete for $Imp(\overline{\mathcal{K}})$, and that $\mathcal{K}$ refutes every implication that is refuted by $\overline{\mathcal{K}}$. *Soundness* of $\mathcal{L}$ is an immediate consequence of the fact that the invariant $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ holds throughout the run of the algorithm. The only difference between the original and the modified algorithm is that the former starts with the empty set of implications whereas the latter starts with a possibly non-empty set $\mathcal{L}_0$ of implications. However, since $\mathcal{L}_0$ is assumed to satisfy $\overline{\mathcal{K}} \subseteq Mod(\mathcal{L}_0)$, the invariant is still satisfied at the start of the modified algorithm.

Because the invariant is satisfied, completeness of $\mathcal{L}$ for $Imp(\overline{\mathcal{K}})$ as well as the fact that $\mathcal{K}$ refutes every implication refuted by $\overline{\mathcal{K}}$ follow by Proposition 2 as soon as we have shown that every implication is decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. Thus, assume that $L \rightarrow R$ is undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$, i.e., it does not follow from $\mathcal{L}$ and is not refuted by $\mathcal{K}$. By Proposition 3, $\mathcal{L}(L) \rightarrow R$ also does not follow from $\mathcal{L}$. In addition, since $L \subseteq \mathcal{L}(L)$, it is also not refuted by $\mathcal{K}$.

We claim that $\mathcal{L}(L)$ is equal to one of the sets $P_i$ considered during the run of the algorithm. In fact, since the final set $P_n = M$ is the lectically largest subset of $M$ and since the lectic order $<$ is total, there is a unique smallest $i$ such that $\mathcal{L}(L) < P_i$. First, assume that $i = 0$. Then $\mathcal{L}(L) < P_0 = \mathcal{L}_0(\emptyset)$. However, $\mathcal{L}_0 \subseteq \mathcal{L}$ implies that $\mathcal{L}(L)$ is as also $\mathcal{L}_0$-closed, which contradicts the fact that $\mathcal{L}_0(\emptyset)$ is the smallest $\mathcal{L}_0$-closed set. If $i > 0$, then $P_{i-1} < L < P_i$, and we can analogously derive a contradiction to the fact the $P_i$ is the lectically next $\mathcal{L}_i$-closed set after $P_{i-1}$.

Thus, let $i$ be such that $\mathcal{L}(L) = P_i$. Then the implication $P_i \rightarrow \mathcal{K}_i(P_i)$ is considered during iteration $i$ of the algorithm. If this implication is not refuted by $\overline{\mathcal{K}}$, then one can show that $R \subseteq \mathcal{K}_i(P_i)$, and use this fact to show that $P_i \rightarrow R$ follows from $\mathcal{L}$, which contradicts our assumption that $L \rightarrow R$ is undecided (see [4] for details).

If $P_i \rightarrow \mathcal{K}_i(P_i)$ is refuted by $\overline{\mathcal{K}}$, then $\mathcal{K}_i$ is extended to a partial context $\mathcal{K}_{i+1}$ that refutes this implication. If $\mathcal{K}_{i+1}$ also refutes $P_i \rightarrow R$, then this implies that $\mathcal{K}$ refutes $L \rightarrow R$, which is again a contradiction (see [4] for details). Otherwise, note that $P_{i+1} = P_i$ and $\mathcal{L}_{i+1} = \mathcal{L}_i$, and thus in the next iteration the expert gets the implication $P_i \to \mathcal{K}_{i+1}(P_i)$. By our assumption, $P_i \to R$ is not refuted by $\mathcal{K}_{i+1}$, and thus $R \subseteq K_{i+1}(P_i)$. In addition, we have $\mathcal{K}_{i+1}(P_i) \subsetneq \mathcal{K}_i(P_i)$ due to the fact that $\mathcal{K}_{i+1}$ refutes $P_i \to \mathcal{K}_i(P_i)$.

If $P_i \to \mathcal{K}_{i+1}(P_i)$ is not refuted by $\overline{\mathcal{K}}$, then we can continue as in the first case above, and derive that $P_i \to R$ follows from $\mathcal{L}$. Otherwise, we can continue as in the second case. However, because in this case the size of the right-hand side of the implication given to the expert strictly decreases, we cannot indefinitely get the second case. This completes our sketch of the proof of Theorem 3.

Termination and correctness of the accordingly modified Algorithm 2 is a trivial consequence of this theorem, i.e., we can also start this algorithm with a non-empty set of background implications $\mathcal{L}_0$ provided that all implications in $\mathcal{L}_0$ are not refuted by $\mathcal{I}$.

## 5 OntoComP: Ontology Completion Plugin for Protégé

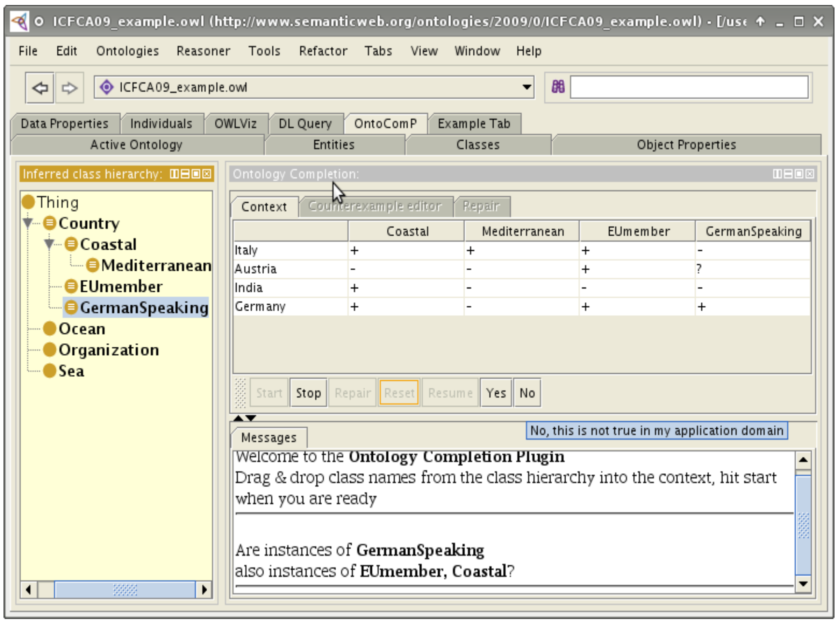
Based on the usability considerations sketched in the previous sections, we have implemented an improved version of our completion algorithm as an open-source tool called OntoComP,[3] which stands for ONTOlogy COMpletion Plugin. It is written in Java as a plugin for the Protégé 4 ontology editor [19]. It communicates with the underlying DL reasoner over the OWL API [8].

OntoComP can easily be integrated into an existing Protégé 4 installation. After installing this plugin, it appears in the Protégé 4 window as a new tab. For completing a knowledge base loaded into Protégé 4, one first needs to classify it with one of the DL reasoners supported by Protégé 4 (e.g., FaCT++ [35] or Pellet [33]). Then one can go to the OntoComP tab to start completion, and create the set $M$ of interesting concepts by dragging and dropping the concept names that are supposed to be in this set from the class hierarchy displayed in the OntoComP tab. Figure 5 displays the OntoComP window during completion of the knowledge base of Example 3.

**Counterexample Generation.** OntoComP has a counterexample editor for supporting the user during counterexample generation. When the user rejects an implication, OntoComP opens a counterexample editor in a new tab, and displays those individuals from the ABox that can potentially be turned into a counterexample by adding assertions for them. Alternatively, the user can introduce a new individual together with assertions that make it a counterexample. During counterexample generation, OntoComP guides the user and notifies her once she has created a valid counterexample to the implication question.

**Error Recovery.** At any point during knowledge base completion, if the user notices that he has made a mistake in one of the previous steps, he can stop the completion and can request to see the history of all answers he has given. OntoComP displays all the questions asked in previous steps, the answers that were

[3] available under `http://code.google.com/p/ontocomp`


ICFCA09_example.owl (http://www.semanticweb.org/ontologies/2009/0/ICFCA09_example.owl) - [/use
File Edit Ontologies Reasoner Tools Refactor Tabs View Window Help
ICFCA09_example.owl
Data Properties Individuals OWLViz DL Query OntoComP Example Tab
Active Ontology Entities Classes Object Properties
Inferred class hierarchy:
Thing
Country
Coastal
Mediterranean
EUmember
GermanSpeaking
Ocean
Organization
Sea
Ontology Completion:
Context Counterexample editor Repair
Start Stop Repair Reset Resume Yes No
No, this is not true in my application domain
Messages

| | Coastal | Mediterranean | EUmember | GermanSpeaking |
|---|---|---|---|---|
| Italy | + | + | + | - |
| Austria | - | - | + | ? |
| India | + | - | - | - |
| Germany | + | - | + | + |

Welcome to the **Ontology Completion Plugin**
Drag & drop class names from the class hierarchy into the context, hit start when you are ready

Are instances of **GermanSpeaking**
also instances of **EUmember, Coastal**?

**Fig. 1.** OntoComP window during completion

given to these questions, and the counterexamples accompanying negative answers. The user can browse the answering history, correct the wrong answers he has given in the previous steps, and can then continue completion. OntoComP then keeps all GCIs and counterexamples that were not marked as incorrect in the knowledge base, and moves all implications to the background knowledge. Pinpointing reasons for consequences (such as inconsistency or unintended subsumption or instance relationships) is not directly integrated into the current version of OntoComP. However, the user could use the pinpointing facilities provided by Protégé 4.

**Deferring Questions.** OntoComP allows the user to defer a question at any point during completion. It achieves this by the approach described in Section 4, i.e., it tries to change the order on $M$ such that a different question is generated as the next question. As already mentioned in Section 4, this need not always succeed.

## 6 Future Work

In addition to further improving and evaluating our completion tool OntoComP, the main topic for future research in this direction will be to 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 a finite 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 the $\mathcal{D}$-concept descriptions that can be built using a finite set of interesting concept and role names. This would result in a notion of completeness where each GCIs that can be built using $\mathcal{D}$ and the given finite set of concept and role names either follows from the TBox (in case it holds in the underlying model) or is refuted by the ABox (in case it does not hold in the underlying model).

The obvious problem caused by this extension is that, in general, the set of attributes becomes infinite, and thus termination of the exploration process is no longer a priori guaranteed. Different extensions of classical attribute exploration (i.e., for full contexts) in this direction are described in [30,31] for the DL $\mathcal{FLE}$, and in [2,3] for the DL $\mathcal{EL}$ and its extension by cyclic concept definitions with greatest fixpoint semantics, $\mathcal{EL}_{\text{gfp}}$. In both approaches, variants of classical attribute exploration are introduced that consider as attributes all concept descriptions built using the given DL and a given finite set of concept and role names. It is shown that the introduced exploration algorithms terminate if the underlying model is finite. We will investigate whether these approaches can be adapted to knowledge base completion.

## References

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Baader, F., Distel, F.: A finite basis for the set of $\mathcal{EL}$-implications holding in a finite model. In: Medina, R., Obiedkov, S. (eds.) ICFCA 2008. LNCS (LNAI), vol. 4933, pp. 46–61. Springer, Heidelberg (2008)
3. Baader, F., Distel, F.: Exploring finite models in the description logic $\mathcal{EL}_{\text{gfp}}$. In: Ferré, S., Rudolph, S. (eds.) ICFCA 2009. LNCS (LNAI), vol. 5548, Springer, Heidelberg (2009)
4. Baader, F., Ganter, B., Sattler, U., Sertkaya, B.: Completing description logic knowledge bases using formal concept analysis. LTCS-Report LTCS-06-02, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany (2006), `http://lat.inf.tu-dresden.de/research/reports.html`
5. Baader, F., Ganter, B., Sertkaya, B., Sattler, U.: Completing description logic knowledge bases using formal concept analysis. In: Proc. of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI 2007), pp. 230–235. AAAI Press, Menlo Park (2007)
6. Baader, F., Peñaloza, R., Suntisrivaraporn, B.: Pinpointing in the description logic $\mathcal{EL}^+$. In: Hertzberg, J., Beetz, M., Englert, R. (eds.) KI 2007. LNCS (LNAI), vol. 4667, pp. 52–67. Springer, Heidelberg (2007)
7. Baader, F., Suntisrivaraporn, B.: Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Proc. of the Int. Conf. on Representing and Sharing Knowledge Using SNOMED (KR-MED 2008), Phoenix, Arizona (2008)
8. Bechhofer, S., Volz, R., Lord, P.: Cooking the semantic web with the OWL API. In: Fensel, D., Sycara, K.P., Mylopoulos, J. (eds.) ISWC 2003. LNCS, vol. 2870, pp. 659–675. Springer, Heidelberg (2003)

9. Burmeister, P., Holzer, R.: On the treatment of incomplete knowledge in formal concept analysis. In: Ganter, B., Mineau, G.W. (eds.) ICCS 2000. LNCS, vol. 1867, pp. 385–398. Springer, Heidelberg (2000)
10. Burmeister, P., Holzer, R.: Treating incomplete knowledge in formal concept analysis. In: Ganter, B., Stumme, G., Wille, R. (eds.) Formal Concept Analysis. LNCS, vol. 3626, pp. 114–126. Springer, Heidelberg (2005)
11. Ganter, B.: Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany (1984)
12. Ganter, B.: Attribute exploration with background knowledge. Theoretical Computer Science 217(2), 215–233 (1999)
13. Ganter, B., Krauße, R.: Pseudo-models and propositional Horn inference. Discrete Applied Mathematics 147(1), 43–55 (2005)
14. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
15. Haarslev, V., Möller, R.: RACER system description. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 701–705. Springer, Heidelberg (2001)
16. Halaschek-Wiener, C., Parsia, B., Sirin, E., Kalyanpur, A.: Description Logic reasoning for dynamic ABoxes. In: Proc. of the 19th Int. Workshop on Description Logics (DL 2006). CEUR-WS, vol. 189 (2006)
17. Holzer, R.: Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part I. Fundamenta Informaticae 63(1), 17–39 (2004)
18. Holzer, R.: Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part II. Fundamenta Informaticae 63(1), 41–63 (2004)
19. Horridge, M., Tsarkov, D., Redmond, T.: Supporting early adoption of OWL 1.1 with Protege-OWL and FaCT++. In: Proc. of the Second Int. Workshop OWL: Experiences and Directions (OWLED 2006). CEUR-WS (2006)
20. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: the making of a web ontology language. Journal of Web Semantics 1(1), 7–26 (2003)
21. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C.: Repairing unsatisfiable concepts in OWL ontologies. In: Sure, Y., Domingue, J. (eds.) ESWC 2006. LNCS, vol. 4011, pp. 170–184. Springer, Heidelberg (2006)
22. Kalyanpur, A., Parsia, B., Sirin, E., Grau, B.C., Hendler, J.A.: Swoop: A web ontology editing browser. Journal of Web Semantics 4(2), 144–153 (2006)
23. Knublauch, H., Fergerson, R.W., Noy, N.F., Musen, M.A.: The Protégé OWL plugin: An open development environment for semantic web applications. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 229–243. Springer, Heidelberg (2004)
24. Meyer, T., Lee, K., Booth, R., Pan, J.Z.: Finding maximally satisfiable terminologies for the description logic $\mathcal{ALC}$. In: Proc. of the 21st National Conf. on Artificial Intelligence (AAAI 2006), pp. 269–274. AAAI Press/The MIT Press (2006)
25. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. Dissertation, Universität Karlsruhe (TH), Germany (2006)
26. Motik, B., Shearer, R., Horrocks, I.: Optimized reasoning in description logics using hypertableaux. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 67–83. Springer, Heidelberg (2007)
27. Oberle, D., Volz, R., Staab, S., Motik, B.: An extensible ontology software environment. In: Handbook on Ontologies, Int. Handbooks on Information Systems, pp. 299–320. Springer, Heidelberg (2004)

28. Obiedkov, S.A.: Modal logic for evaluating formulas in incomplete contexts. In: Priss, U., Corbett, D.R., Angelova, G. (eds.) ICCS 2002. LNCS, vol. 2393, pp. 314–325. Springer, Heidelberg (2002)
29. Reeg, S., Weiß, W.: Properties of Finite Lattices. Diplomarbeit, TH Darmstadt, Germany (1990)
30. Rudolph, S.: Exploring relational structures via $\mathcal{FLE}$. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) ICCS 2004. LNCS, vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
31. Rudolph, S.: Relational Exploration: Combining Description Logics and Formal Concept Analysis for Knowledge Specification. Ph.D. Dissertation, Fakultät Mathematik und Naturwissenschaften, TU Dresden, Germany (2006)
32. Schlobach, S., Cornet, R.: Non-standard reasoning services for the debugging of description logic terminologies. In: Proc. of the Eighteenth Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 355–362. Morgan Kaufmann, San Francisco (2003)
33. Sirin, E., Parsia, B.: Pellet: An OWL DL reasoner. In: Proc. of the 2004 Int. Workshop on Description Logics (DL 2004). CEUR Workshop Proc., vol. 104. CEUR-WS.org (2004)
34. Stumme, G.: Attribute exploration with background implications and exceptions. In: Data Analysis and Information Systems. Statistical and Conceptual approaches. Proc. of GfKl 1995. Studies in Classification, Data Analysis, and Knowledge Organization, vol. 7, pp. 457–469. Springer, Heidelberg (1996)
35. Tsarkov, D., Horrocks, I.: FaCT++ description logic reasoner: System description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
36. Wille, R.: Restructuring lattice theory: An approach based on hierarchies of concepts. In: Ordered Sets, pp. 445–470. Reidel, Dordrecht (1982)
37. Wolstencroft, K., Brass, A., Horrocks, I., Lord, P.W., Sattler, U., Turi, D., Stevens, R.: A little semantic web goes a long way in biology. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 786–800. Springer, Heidelberg (2005)