TECHNISCHE UNIVERSITÄT DRESDEN

Dresden University of Technology
Institute for Theoretical Computer Science
Chair for Automata Theory

LTCS–Report

Completing Description Logic Knowledge Bases using Formal Concept Analysis

F. Baader B. Ganter U. Sattler B. Sertkaya

LTCS-Report 06-02

Lehrstuhl für Automatentheorie
Institut für Theoretische Informatik
TU Dresden
http://lat.inf.tu-dresden.de

Hans-Grundig-Str. 25
01062 Dresden
Germany

Completing Description Logic Knowledge Bases using Formal Concept Analysis

Franz Baader
Institute for Theoretical Computer Science
TU Dresden, Germany
baader@tcs.inf.tu-dresden.de

Bernhard Ganter
Institute for Algebra
TU Dresden, Germany
ganter@math.tu-dresden.de

Ulrike Sattler
Department of Computer Science
University of Manchester, UK
Ulrike.Sattler@manchester.ac.uk

Barış Sertkaya
Institute for Theoretical Computer Science
TU Dresden, Germany
sertkaya@tcs.inf.tu-dresden.de

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 assertional part 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 sense.

Contents

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 the adoption of the DL-based language OWL [12] as standard ontology language for the semantic web. As a consequence of this standardization, many ontology editors support OWL [2, 15, 17, 13], and ontologies written in OWL are employed in more and more applications. As the size of such ontologies grows, tools that support improving the quality of large DL-based ontologies 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 first 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 [24, 22, 23, 20, 14]. 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 want to develop 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 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 [27]. 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 certain 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 human protein phosphatases may require the biologist to study the relevant literature, query existing protein databases, or even to carry out new experiments. Thus, new biological knowledge may be acquired by the expert in the process.

Attribute exploration [6] is an approach developed in Formal Concept Analysis (FCA) [7] 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 [26], 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 providing 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 the ABox and a concept C occurring in the TBox. If we cannot deduce from the TBox and ABox 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 the TBox and ABox imply that i is an instance of $\neg C$. Thus, our knowledge about the relationships between individuals and concepts is incomplete: if TBox and ABox 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 [3, 18, 4]. 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 first briefly review some notions and results from FCA. Then, we develop our variant of FCA that can deal with partial contexts, and finally describe an attribute exploration procedure that works with partial contexts. In Section 3, we give a brief introduction into description logics, show how a DL knowledge base gives rise to a partial context, and specialize our new attribute exploration procedure to the case of partial contexts induced by DL knowledge bases. In Section 4, we describe a first experimental implementation of a tool for completing DL knowledge bases, and in Section 5 we summarize the results of the paper and mention some topics for future research.

2 Formal Concept Analysis

In the first part of this section, we briefly recall some notions and results from classical formal concept analysis. More details and proofs of the results that we mention can be found in [7]. In the second part, we introduce our extension to the case of partial knowledge, and in the third part we develop a variant of attribute exploration that works for partial knowledge.

2.1 The classical case

Formal Concept Analysis (FCA) [7] is a field of applied mathematics that is based on a lattice-theoretic formalization of the notions of a concept and of a hierarchy of concepts. It is supposed to facilitate the use of mathematical reasoning for conceptual data analysis and knowledge processing. In FCA, one represents data in the form of a *formal context*, which in its simplest form is a way of specifying which attributes (properties) are satisfied by which objects (individuals). Formally, a formal context is defined as follows:

Definition 2.1 *A* formal context *is a triple* $\mathbb{K} = (G, M, I)$*, where* G *is a set of objects,* M *is a set of attributes, and* $I \subseteq G \times M$ *is a relation that associates each object* g *with the attributes satisfied by* g*. In order to express that an object* g *is in relation* I *with an attribute* m*, we write* gIm*.*

A formal context is usually visualised as a crosstable, where the rows represent the objects, and the columns represent the attributes. A cross in column m of row g means that object g has attribute m, and the absence of a cross means that g does not have attribute m. In this paper, we will always assume the set of attributes M to be *finite*.

Let $\mathbb{K} = (G, M, I)$ be a formal context. For a set of objects $A \subseteq G$, the *intent* A' of A is the set of attributes that are satisfied by all objects in A, i.e.,

$$A' := \{p \in M \mid \forall a \in A\colon aIp\}.$$

Similarly, for a set of attributes $B \subseteq M$, the *extent* B' of B is the set of objects that satisfy all attributes in B, i.e.,

$$B' := \{o \in G \mid \forall b \in B\colon oIb\}.$$

It is easy to see that, for $A_1 \subseteq A_2 \subseteq G$ (resp. $B_1 \subseteq B_2 \subseteq M$), we have

- $A_2' \subseteq A_1'$ (resp. $B_2' \subseteq B_1'$),
- $A_1 \subseteq A_1''$ and $A_1' = A_1'''$ (resp. $B_1 \subseteq B_1''$ and $B_1' = B_1'''$).

As an easy consequence one obtains that the $\cdot''$ operation is a *closure operator* on both G and M.

Definition 2.2 *Let* S *be a set and* φ *a mapping from the powerset of* S *into itself. Then* φ *is called a* closure operator *on* S *if it is*

- extensive: $B \subseteq \varphi(B)$ *for all* $B \subseteq S$*;*
- monotone: $B_1 \subseteq B_2$ *implies* $\varphi(B_1) \subseteq \varphi(B_2)$*; and*
- idempotent: $\varphi(\varphi(B)) = \varphi(B)$*.*

We say that a set $B \subseteq S$ *is* φ*-closed if* $B = \varphi(B)$*.*

Given a formal context, one common method to analyse it is to find (a base of) the implications between the attributes of this context. Implications between attributes are constraints between attributes that hold in the given context. They are statements of the form

> "Every object that satisfies the attributes $m_{i1}, \ldots, m_{ik}$ also satisfies the attributes $m_{j1}, \ldots, m_{j\ell}$."

Formally, an implication between attributes is defined as follows:

Definition 2.3 *Let $\mathbb{K} = (G, M, I)$ be a formal context. An* implication *between the attributes in M is a pair of sets $L, R \subseteq M$, usually written as $L \to R$. An implication $L \to R$* holds *in $\mathbb{K}$ if every object of $\mathbb{K}$ that has all of the attributes in L also has all of the attributes in R, i.e., if $L' \subseteq R'$. We denote the set of implications that hold in $\mathbb{K}$ by Imp$(\mathbb{K})$.*

It is easy to see that an implication $L \to R$ holds in $\mathbb{K}$ iff R is contained in the $\cdot''$-closure of L, i.e., if $R \subseteq L''$.

A set of implications induces its own closure operator.

Definition 2.4 *Let $\mathcal{L}$ be a set of implications. For 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*
- *$L_i \to R_i \in \mathcal{L}$ and $L_i \subseteq Q$ imply $R_i \subseteq Q$.*

It is easy to see that $\mathcal{L}(\cdot)$ is indeed a closure operator.

From a logician's point of view, computing the implication closure of a set of attributes P is just computing consequences in propositional Horn logic. In fact, the notions we have just defined can easily be reformulated in propositional logic. To this purpose, we view the attributes as propositional variables. An implication $L \to R$ can then be expressed by the formula $\phi_{L\to R} := \bigwedge_{\ell \in L} \ell \to \bigwedge_{r \in R} r$. Let $\Gamma_{\mathcal{L}}$ be the set of formulae corresponding to the set of implications $\mathcal{L}$. Then

$$\mathcal{L}(P) = \{b \in M \mid \Gamma_{\mathcal{L}} \cup \{\bigwedge_{p \in P} p\} \models b\},$$

where $\models$ stands for classical propositional consequence. Obviously, the formulae in $\Gamma_{\mathcal{L}}$ are Horn clauses. For this reason, the implication closure $\mathcal{L}(B)$ of a set of attributes B can be computed in time linear in the size of $\mathcal{L}$ and B using methods for deciding satisfiability of sets of propositional Horn clauses [5]. Alternatively, these formulae can be viewed as expressing functional dependencies in relational database, and thus the linearity result can also be obtained by using methods for deriving new functional dependencies from given ones [16].

Definition 2.5 *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.*

Again, the consequence operation between implications coincides with the usual logical notion of consequence if one translates implications into Horn clauses, as described above.

If $\mathcal{J}$ is sound and complete for $Imp(\mathbb{K})$, then the two closure operators that we have introduced until now coincide, i.e., $B'' = \mathcal{J}(B)$ for all $B \subseteq M$. Consequently, given a base $\mathcal{J}$ for $Imp(\mathbb{K})$, any question of the form "$B_1 \to B_2 \in Imp(\mathbb{K})$?" can be answered in time linear in the size of $\mathcal{J} \cup \{B_1 \to B_2\}$ since it is equivalent to asking whether $B_2 \subseteq B_1'' = \mathcal{J}(B_1)$.

In many applications, one needs to classify a large (or even infinite) set of objects with respect to a relatively small set of attributes. Moreover, it is often the case that the formal context is not given explicitly as a crosstable, but it is rather "known" to a domain expert. In such cases, Ganter's interactive *attribute exploration* algorithm [6] has proved to be a useful method to efficiently capture the expert's knowledge. By asking implication questions to a domain expert, the method computes a base for $Imp(\mathbb{K})$ and a subcontext $\mathbb{K}'$ of the $\mathbb{K}$ such that $Imp(\mathbb{K}') = Imp(\mathbb{K})$. For each implication question, the expert either says that it holds in $\mathbb{K}$, in which case the implication is added to the base, or the expert gives a counterexample from $\mathbb{K}$, which is then added to $\mathbb{K}'$.

In order to produce a base for $Imp(\mathbb{K})$, one could, of course, enumerate all possible implications, and have the expert decide for each of them whether it holds in $\mathbb{K}$ or not. Obviously, this would be very inefficient, and produce all of $Imp(\mathbb{K})$ rather than a small base for this set. The main idea underlying *attribute exploration* (see Algorithm 1) is that one can restrict the attention to implications having a left-hand side that is closed under the implications of the context, and whose right-hand side is obtained from the left-hand side by applying the $\cdot''$ closure operator. The left-hand sides are enumerated in a certain order, called the *lectic order*, which ensures that it is sufficient to build the implication closure w.r.t. the already computed implications. In addition, the $\cdot''$ operator is computed w.r.t. the already computed subcontex rather than the full context $\mathbb{K}$.

Definition 2.6 *Assume that $M = \{m_1, \ldots, m_n\}$ and fix some linear order $m_1 < m_2 < \cdots m_n$ on M. This order imposes a linear order on the power set of M,*

Algorithm 1 Attribute exploration

Initialization
$\mathbb{K}_0$ {initial formal context, possibly empty set of objects}
$\mathcal{L}_0 := \varnothing$ {initial empty set of implications}
$P_0 := \varnothing$ {lectically smallest $\mathcal{L}_0$-closed subset of M}
$i := 0$
while $P_i \neq M$ **do**
  Compute P_i'' w.r.t. $\mathbb{K}_i$
  if $P_i \neq P_i''$ **then**
    Ask the expert if $P_i \rightarrow P_i''$ holds
    if yes **then**
      $\mathbb{K}_{i+1} := \mathbb{K}_i$
      $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \rightarrow P_i'' \setminus P_i\}$
      $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P_i <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
    else
      Get an object o of $\mathbb{K}$ from the expert s.t: $P_i \subseteq o'$ and $P_i'' \not\subseteq o'$
      $\mathbb{K}_{i+1} := \mathbb{K}_i \cup \{o\}$
      $P_{i+1} := P_i$
      $\mathcal{L}_{i+1} := \mathcal{L}_i$
    end if
  else
    $\mathbb{K}_{i+1} := \mathbb{K}_i$
    $\mathcal{L}_{i+1} := \mathcal{L}_i$
    $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P_i <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
  end if
  $i := i + 1$
end while

called the lectic order*, which we also denote by $<$: For $m_i \in M$ and $A, B \subseteq M$ we define*

$$A <_i B \quad \text{iff} \quad m_i \in B, m_i \notin A \text{ and } \forall j < i.\ (m_j \in A \Leftrightarrow m_j \in B).$$

The order $<$ is the union of these orders $<_i$, i.e.,

$$A < B \quad \text{iff} \quad A <_i B \text{ for some } i \in M.$$

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

The following proposition shows how one can enumerate all closed sets w.r.t. a given closure operator in the lectic order.

Proposition 2.7 *Given a closure operator φ on M and a φ-closed set $A \subsetneq M$, the next φ-closed set following A in the lectic order is*

$$\varphi((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$$

where j is maximal such that $A <_j \varphi((A \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\}$.

It can be shown that Algorithm 1 always terminates, and that the set of implications $\mathcal{L}_i$ obtained after termination is a base for $Imp(\mathbb{K})$. More precisely, one can show that it is the so-called *Duquenne-Guigues* base of the context, which contains a minimal number of implications. This base can be described independently of the algorithm, based on the notion of a *pseudo-intent* of the context.

Definition 2.8 *A set $P \subseteq M$ is called a pseudo-intent of the context $\mathbb{K} = (G, M, I)$ if $P \neq P''$ and $Q'' \subseteq P$ holds for all pseudo-intents $Q \subsetneq P$.*

The Duquenne-Guigues base of $\mathbb{K}$ consists of implications that have the pseudo-intents of $\mathbb{K}$ as left-hand sides.

Definition 2.9 *The* Duquenne-Guigues *base of the context $\mathbb{K}$ consists of the implications $P \to P'' \setminus P$, where P ranges over all pseudo-intents of $\mathbb{K}$.*

2.2 Partial contexts

The goal of this subsection is to extend the classical approach to FCA described above 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. As above, we assume that we have a finite set M of attributes and a (possibly infinite) set of objects.

Definition 2.10 *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*.*

Note that the notion of a full context introduced in this definition coincides with the notion of a formal context introduced in the previous section: a set of fods $\overline{\mathcal{K}}$ corresponds to the formal context $\mathbb{K}_{\overline{\mathcal{K}}} := (\overline{\mathcal{K}}, M, I)$, where $(\overline{A}, \overline{S}) I m$ iff $m \in \overline{A}$ for all $(\overline{A}, \overline{S}) \in \overline{\mathcal{K}}$.

A partial context can be extended by either adding new pods or by extending existing pods.

Definition 2.11 *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 extend the definition of the implications of a formal context to the case of partial contexts.

Definition 2.12 *Let* $L, R \subseteq M$*. The implication* $L \to R$ *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* $Mod(\mathcal{L})$*.*

If (A, S) is a fod and $L \to R$ an implication, then (A, S) does not refute $L \to R$ iff $L \subseteq A$ implies $R \cap S = \varnothing$ iff $L \subseteq A$ implies $R \subseteq M \setminus S = A$. Thus, the implication $L \to R$ is not refuted by the full context $\overline{\mathcal{K}}$ iff it holds in the corresponding formal context $\mathbb{K}_{\overline{\mathcal{K}}}$.

The following simple facts regarding the connection between $Imp(\cdot)$, $Mod(\cdot)$, and the consequence operator for implications will be employed later on without explicitly mentioning their application:

- If $\overline{\mathcal{K}}$ is a full context and $\mathcal{L}$ a set of implications, then $\overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ iff $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$.
- If $\mathcal{K}$ is a partial context and $\mathcal{L}$ a set of implications, then $\mathcal{L} \subseteq Imp(\mathcal{K})$ implies that every implication that follows from $\mathcal{L}$ belongs to $Imp(\mathcal{K})$.

The following is a trivial fact regarding the connection between partial contexts and the implications they do not refute.

Proposition 2.13 *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* $\mathcal{K}$*.*

The following facts are immediate consequences of the definition of $\mathcal{K}(\cdot)$:

- If $P \subseteq Q$, then $\mathcal{K}(P) \subseteq \mathcal{K}(Q)$.

- If $\mathcal{K} \leq \mathcal{K}'$, then $\mathcal{K}'(P) \subseteq \mathcal{K}(P)$.

For a full context $\overline{\mathcal{K}}$, the operator $\overline{\mathcal{K}}(\cdot)$ coincides with the $\cdot''$ operator of the corresponding formal context $\mathbb{K}_{\overline{\mathcal{K}}}$. In fact, if $\mathcal{L}$ is a base for $Imp(\mathbb{K}_{\overline{\mathcal{K}}})$, then we have $m \in P''$ iff $m \in \mathcal{L}(P)$ iff $P \to \{m\}$ follows from $\mathcal{L}$ iff $P \to \{m\}$ holds in $\mathbb{K}_{\overline{\mathcal{K}}}$ iff $P \to \{m\}$ is not refuted by $\overline{\mathcal{K}}$ iff $m \in \overline{\mathcal{K}}(P)$.

The following proposition connects refutation by a partial context to refutation by the realizers of this partial context.

Proposition 2.14 *Let $\mathcal{K}$ be a partial context. An implication is refuted by $\mathcal{K}$ iff it is refuted by all realizers of $\mathcal{K}$.*

Proof. First, let $L, R \subseteq M$ be such that $L \to R$ is refuted by $\mathcal{K}$, and let $\overline{\mathcal{K}}$ be a realizer of $\mathcal{K}$. Then, by the definition of refutation, there is an $(A, S) \in \mathcal{K}$ such that $L \subseteq A$ and $R \cap S \neq \varnothing$, and by the definition of a realizer, there is a fod $(\overline{A}, \overline{S}) \in \overline{\mathcal{K}}$ such that $A \subseteq \overline{A}$ and $S \subseteq \overline{S}$. Obviously, we have $L \subseteq \overline{A}$ and $R \cap \overline{S} \neq \varnothing$. Thus, $L \to R$ is refuted by $\overline{\mathcal{K}}$ as well.

Second, assume the implication $L \to R$ is not refuted by $\mathcal{K}$, i.e., for every pod $(A, S) \in \mathcal{K}$ we have that $L \subseteq A$ implies $R \cap S = \varnothing$. We define a realizer $\overline{\mathcal{K}}$ of $\mathcal{K}$ as follows. Consider a pod $(A, S) \in \mathcal{K}$. If $L \not\subseteq A$, then we add $(A, M \setminus A)$ to $\overline{\mathcal{K}}$: obviously, $(A, M \setminus A)$ realizes (A, S) and does not refute $L \to R$. If $L \subseteq A$, then we also have $R \cap S = \varnothing$, and we add $(M \setminus S, S)$ to $\overline{\mathcal{K}}$: obviously, $(M \setminus S, S)$ realizes (A, S) and does not refute $L \to R$. □

Note that the if-direction of this proposition need not hold if we consider a set of implications rather than a single implication. For example, consider the implications $\{a, b\} \to \{c\}, \{a\} \to \{b\}$. The partial context that consists of the single pod $(\{a\}, \{c\})$ does not refute any of these two implications, but each realizer of this partial context refutes one of them.

In the proof of the only-if-direction, we did not make use of the fact that $\overline{\mathcal{K}}$ is a full context. Thus, this direction also holds for partial contexts.

Lemma 2.15 *If $\mathcal{K}, \mathcal{K}'$ are partial contexts such that $\mathcal{K} \leq \mathcal{K}'$, then every implication refuted by $\mathcal{K}$ is also refuted by $\mathcal{K}'$.*

2.3 Attribute exploration with partial contexts

In contrast to existing work on extending FCA to the case of partial knowledge [3, 18, 4], we do *not* assume that the expert has only partial knowledge and thus cannot answer all implication questions. In principle, our expert is assumed to have access to a full context $\overline{\mathcal{K}}$ and thus can answer all implication questions w.r.t. $\overline{\mathcal{K}}$.[1] What is partial is the subcontext that the attribute exploration algorithm works with. The reason is that the initial context may be partial, and the same is true for the counterexamples that the experts provides for implications that do not hold in $\overline{\mathcal{K}}$.

More formally, 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 $L \to R$ (and thus $L \to R$ holds in the corresponding formal context $\mathbb{K}_{\overline{\mathcal{K}}}$). 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 2.16 *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 all undecided implications by either adding the implication to $\mathcal{L}$ or extending $\mathcal{K}$ such that it refutes the implication. If all implications are decided, then our goal is achieved.

Proposition 2.17 *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 $\mathit{Imp}(\overline{\mathcal{K}})$.*

Proof. First, assume that there is an implication $L \to R$ in $\mathit{Imp}(\overline{\mathcal{K}})$ that does not follow from $\mathcal{L}$. By our assumption, $L \to R$ is decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$, and thus it is refuted by $\mathcal{K}$. However, according to Proposition 2.14, it is then also refuted by the realizer $\overline{\mathcal{K}}$ of $\mathcal{K}$, which contradicts our assumption that $L \to R$ belongs to $\mathit{Imp}(\overline{\mathcal{K}})$.

Second, assume that $L \to R$ is an implication that is refuted by $\overline{\mathcal{K}}$, but is not refuted by $\mathcal{K}$. Since $L \to R$ is decided, this implies that $L \to R$ follows from $\mathcal{L}$.

[1] though finding these answers may involve literature study, or even proving new mathematical theorems or carrying out new experiments.

However, $\overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ implies $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$, and thus $L \rightarrow R$ also belongs to $Imp(\overline{\mathcal{K}})$. This contradicts our assumption that $L \rightarrow R$ is refuted by $\overline{\mathcal{K}}$. $\square$

How can we find the undecided 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 2.18 *Let $\mathcal{L}$ be a set of implications and $L \rightarrow R$ an implication. Then, $L \rightarrow R$ follows from $\mathcal{L}$ iff $\mathcal{L}(L) \rightarrow R$ follows from $\mathcal{L}$.*

Given an $\mathcal{L}$-closed set L as left-hand side, what kind of right-hand sides should we consider? Obviously, we need not consider right-hand sides R for which the implication $L \rightarrow R$ is refuted by $\mathcal{K}$: such implications are already decided. By Proposition 2.13, the largest right-hand side R such that $L \rightarrow R$ is not refuted by $\mathcal{K}$ is $R = \mathcal{K}(L)$. It is actually enough to consider just this right-hand side. In fact, once we have decided $L \rightarrow \mathcal{K}(L)$ (by either extending $\mathcal{K}$ such that it refutes the implication or adding the implication to $\mathcal{L}$), all implications $L \rightarrow R'$ with $R' \subseteq \mathcal{K}(L)$ are also decided.

In order to enumerate all left-hand sides, we again use the lectic order and the procedure derived from Proposition 2.7 for enumerating all $\mathcal{L}$-closed sets w.r.t. this order.

Until now, we have talked as if there was a fixed set of implications $\mathcal{L}$ and a fixed partial context $\mathcal{K}$ to work with. In reality, however, both $\mathcal{L}$ and $\mathcal{K}$ are changed during the run of our procedure. We start with an empty set of implications and an initial partial context, and the procedure can extend both. The following proposition shows that the left-hand sides of the previously added implications are also closed with respect to the extended set of implications. This is due to the fact that the left-hand sides are enumerated in lectic order.

Proposition 2.19 *Let $\mathcal{L}$ be a set of implications and $P_1 < \ldots < P_n$ the lectically first n $\mathcal{L}$-closed sets. If $\mathcal{L}$ is extended with $L \rightarrow R$ s.t. L is $\mathcal{L}$-closed and $P_n < L$, then $P_1, \ldots, P_n$ are still the lectically first n closed sets with respect to the extended set of implications.*

Proof. If $P_1 < \ldots < P_n$ and $P_n < L$, then $P_i < L$ for $i = 1, \ldots, n$ by transitivity of $<$. Since $<$ is irreflexive and contains the strict subset order, $L \not\subseteq P_i$ holds for $i = 1, \ldots, n$. Consequently, the $\mathcal{L}$-closed sets P_i are closed w.r.t. $L \rightarrow R$, and thus also w.r.t. the extended set of implications $\mathcal{L}' := \mathcal{L} \cup \{L \rightarrow R\}$.

It remains to show that $P_1, \ldots, P_{n-1}$ are all the $\mathcal{L}'$-closed sets smaller than P_n. Thus, assume that $P < P_n$ is an $\mathcal{L}'$-closed set. Since $\mathcal{L} \subseteq \mathcal{L}'$, we know that P is also $\mathcal{L}$-closed, and thus it is actually one of the sets P_i, $1 \leq i < n$. $\square$

If an implication has been added because the expert has stated that it holds in $\overline{\mathcal{K}}$, then we can extend the current context $\mathcal{K}$ by applying the implications to the first component of every pod in $\mathcal{K}$. To be more precises, for a partial context $\mathcal{K}$ and a set of implications $\mathcal{L}$ we define

$$\mathcal{L}(\mathcal{K}) := \{(\mathcal{L}(A), S) \mid (A, S) \in \mathcal{K}\}.$$

The following is a simple consequence of this definition.

Proposition 2.20 *Let $\mathcal{K} \leq \overline{\mathcal{K}}$ be a partial and a full context, respectively, and let $\mathcal{L}$ be a set of implications such that $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$. Then $\mathcal{L}(\mathcal{K})$ is a partial context and $\mathcal{K} \leq \mathcal{L}(\mathcal{K}) \leq \overline{\mathcal{K}}$.*

Proof. Obviously, $\mathcal{K} \leq \mathcal{L}(\mathcal{K})$ follows from the fact that $A \subseteq \mathcal{L}(A)$. To show $\mathcal{L}(\mathcal{K}) \leq \overline{\mathcal{K}}$, we consider a pod $(A, S) \in \mathcal{K}$. We must show that $(\mathcal{L}(A), S)$ is realized by some fod in $\overline{\mathcal{K}}$. We know that (A, S) is realized by some fod in $\overline{\mathcal{K}}$, i.e., there is a fod $(\overline{A}, \overline{S}) \in \overline{\mathcal{K}}$ such that $A \subseteq \overline{A}$ and $S \subseteq \overline{S}$. Since $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$, we have $\mathcal{L}(\overline{A}) = \overline{A}$, and thus $\mathcal{L}(A) \subseteq \mathcal{L}(\overline{A}) = \overline{A}$. This shows that $(\overline{A}, \overline{S})$ also realizes $(\mathcal{L}(A), S)$.

The fact that $\mathcal{L}(\mathcal{K})$ is a partial context, i.e., that $\mathcal{L}(A) \cap S = \varnothing$ holds for all $(A, S) \in \mathcal{K}$, is an immediate consequence of $\mathcal{L}(\mathcal{K}) \leq \overline{\mathcal{K}}$. □

Going from $\mathcal{K}$ to $\mathcal{L}(\mathcal{K})$ is actually only one way to extend the current context based on the already computed implications. For example, if we have the pod $(\{\ell\}, \{n\})$ and the implication $\{\ell, m\} \rightarrow \{n\}$ is not refuted by $\overline{\mathcal{K}}$, then we know that m must belong to the second component of every fod realizing $(\{\ell\}, \{n\})$. Consequently, we can extend $(\{\ell\}, \{n\})$ to $(\{\ell\}, \{m, n\})$. To allow also 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. The following proposition shows that the right-hand sides of implications accepted by the expert and computed with respect to the smaller partial context are identical to the ones that would have been computed with respect to the extended one.

Proposition 2.21 *Let $\mathcal{K} \leq \mathcal{K}' \leq \overline{\mathcal{K}}$, where $\mathcal{K}, \mathcal{K}'$ are partial contexts and $\overline{\mathcal{K}}$ is a full context. If $L \rightarrow \mathcal{K}(L)$ is an implication that is not refuted by $\overline{\mathcal{K}}$, then $L \rightarrow \mathcal{K}(L)$ is not refuted by $\mathcal{K}'$ and $\mathcal{K}(L) = \mathcal{K}'(L)$.*

Proof. We have $\mathcal{K}' \leq \overline{\mathcal{K}}$, and thus Proposition 2.14 implies that $L \rightarrow \mathcal{K}(L)$ is not refuted by $\mathcal{K}'$. Since $\mathcal{K} \leq \mathcal{K}'$, we have $\mathcal{K}'(L) \subseteq \mathcal{K}(L)$. If this inclusion were strict, then $L \to \mathcal{K}(L)$ would be refuted by $\mathcal{K}'$ by Proposition 2.13. Thus, we have shown that $\mathcal{K}(L) = \mathcal{K}'(L)$. □

Based on these considerations, our attribute exploration algorithm for partial contexts is described in Algorithm 2. The following proposition shows that this

Algorithm 2 Attribute exploration for partial contexts

Initialization
$\mathcal{K}_0$ {initial partial context, realized by the underlying full context $\overline{\mathcal{K}}$}
$\mathcal{L}_0 := \varnothing$ {initial empty set of implications}
$P_0 := \varnothing$ {lectically smallest $\mathcal{L}_0$-closed subset of M}
$i := 0$
while $P_i \neq M$ **do**
  Compute $\mathcal{K}_i(P_i)$
  if $P_i \neq \mathcal{K}_i(P_i)$ **then** {$P_i \to \mathcal{K}_i(P_i)$ is undecided}
    Ask the expert if the undecided implication $P_i \to \mathcal{K}_i(P_i)$ is refuted by $\overline{\mathcal{K}}$
    if no **then** {$P_i \to \mathcal{K}_i(P_i)$ not refuted}
      $\mathcal{K}_{i+1} := \mathcal{K}'$ where $\mathcal{K}'$ is a partial context such that $\mathcal{K}_i \leq \mathcal{K}' \leq \overline{\mathcal{K}}$
      $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \to \mathcal{K}_i(P_i) \setminus P_i\}$
      $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P_i <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
    else {$P_i \to \mathcal{K}_i(P_i)$ refuted}
      Get a partial context $\mathcal{K}'$ from the expert such that $\mathcal{K}_i \leq \mathcal{K}' \leq \overline{\mathcal{K}}$ and $P_i \to \mathcal{K}_i(P_i)$ is refuted by $\mathcal{K}'$
      $\mathcal{K}_{i+1} := \mathcal{K}'$
      $P_{i+1} := P_i$
      $\mathcal{L}_{i+1} := \mathcal{L}_i$
    end if
  else {trivial implication}
    $\mathcal{K}_{i+1} := \mathcal{K}_i$
    $\mathcal{L}_{i+1} := \mathcal{L}_i$
    $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P_i <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
  end if
  $i := i + 1$
end while

algorithm always terminates, and in which sense it is correct.

Proposition 2.22 *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 2 terminates, and upon termination it outputs a partial context $\mathcal{K}$ and a set of implications $\mathcal{L}$ such that*

- *$\mathcal{L}$ is sound and complete for $Imp(\overline{\mathcal{K}})$, and*
- *$\mathcal{K}$ refutes every implication that is refuted by $\overline{\mathcal{K}}$.*

Proof. First, we show termination. The algorithm starts with the lectically smallest $\mathcal{L}_0$-closed set $P_0 = \mathcal{L}_0(\varnothing)$. At each execution of the while loop, it performs one of the following operations:

1. it extends the current set of implications $\mathcal{L}_i$, and continues with the lectically next closed set P_{i+1} computed by using the extended set of implications $\mathcal{L}_{i+1}$ (lines 12,13 in Algorithm 2).
2. it extends the current context $\mathcal{K}_i$ to a context $\mathcal{K}_{i+1}$ that does not refute any of the implications in $\mathcal{L}_i$, and continues with $P_{i+1} := P_i$ (lines 16,17).
3. it continues with the lectically next closed set P_{i+1}, computed by using the current set of implications $\mathcal{L}_i$ (line 23).

Steps of the form 1 or 3 can be executed only finitely often. In fact, in each of these steps, a lectically larger set is generated. Since M is finite, there are only finitely many subsets of M, and thus every strictly ascending chain w.r.t. $<$ is obviously finite. In steps of the form 2, the algorithm continues with $P_{i+1} := P_i$, but extends $\mathcal{K}_i$ to a partial context $\mathcal{K}_{i+1}$ that refutes the implication $P_i \to \mathcal{K}_i(P_i)$. Consequently, $\mathcal{K}_{i+1}(P_i) \subsetneq \mathcal{K}_i(P_i)$. This shows that, for a fixed set P_i, steps of the form 2 can also be applied only finitely often. Thus, we have shown that termination is guaranteed.

Second, to show soundness of the output set of implications $\mathcal{L}$ for $Imp(\overline{\mathcal{K}})$, it is sufficient to note that the invariant $\mathcal{K}_i \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L}_i)$ is preserved throughout the run of the algorithm. Consequently, we also have $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$. But then $\overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$ implies $\mathcal{L} \subseteq Imp(\overline{\mathcal{K}})$, and thus soundness of $\mathcal{L}$ for $Imp(\overline{\mathcal{K}})$.

Third, since we have $\mathcal{K} \leq \overline{\mathcal{K}} \subseteq Mod(\mathcal{L})$, Proposition 2.17 shows that completeness of $\mathcal{L}$ for $Imp(\mathcal{K})$ as well as the fact that $\mathcal{K}$ refutes every implication that is refuted by $\overline{\mathcal{K}}$ follow as soon as we have shown that every implication is decided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. To see this, consider the sets $P_0 = \mathcal{L}_0(\varnothing), P_1, \ldots, P_n = M$ generated during the run of the algorithm. We have $P_0 < P_1 < \ldots < P_n$, and iterated applications of Proposition 2.19 show that $P_0, P_1, \ldots, P_n$ are all the $\mathcal{L}$-closed subsets of M.

Now, assume that the implication $L \to R$ is undecided w.r.t. $\mathcal{K}$ and $\mathcal{L}$. Thus, $L \to R$ does not follow from $\mathcal{L}$ and is not refuted by $\mathcal{K}$. By Proposition 2.18, $\mathcal{L}(L) \to R$ also does not follow from $\mathcal{L}$. In addition, since $L \subseteq \mathcal{L}(L)$, it is also not refuted by $\mathcal{K}$. Since $\mathcal{L}(L)$ is $\mathcal{L}$-closed, there is an i such that $\mathcal{L}(L) = P_i$. During iteration i of the algorithm, the implication $P_i \to \mathcal{K}_i(P_i)$ is considered.

First, assume that this implication is not refuted by $\overline{\mathcal{K}}$. Then, $P_i \to \mathcal{K}_i(P_i)$ follows from $\mathcal{L}_{i+1}$, and thus also from its superset $\mathcal{L}$. However, the fact that $P_i \to R$ is not refuted by $\mathcal{K}$ implies that it is also not refuted by $\mathcal{K}_i$ since $\mathcal{K}_i \leq \mathcal{K}$ (Lemma 2.15). Thus $R \subseteq \mathcal{K}_i(P_i)$ by Proposition 2.13, and the fact that $P_i \to \mathcal{K}_i(P_i)$ follows from $\mathcal{L}$ implies that $P_i \to R$ follows from $\mathcal{L}$, which yields a contradiction.

Second, assume that $P_i \to \mathcal{K}_i(P_i)$ is refuted by $\overline{\mathcal{K}}$. Then, $\mathcal{K}_i$ is extended to a partial context $\mathcal{K}_{i+1}$ that refutes the implication $P_i \to \mathcal{K}_i(P_i)$. If $\mathcal{K}_{i+1}$ also refutes $P_i \to R$, then we are done since $\mathcal{K}_{i+1} \leq \mathcal{K}$ implies that also $\mathcal{K}$ refutes $P_i \to R$, and thus $\mathcal{K}$ refutes $L \to R$ because $L \subseteq P_i$. 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 shows that, eventually, the implication $L \to R$ will become decided w.r.t. some $\mathcal{K}_j$ and $\mathcal{L}_j$ for some $j \geq i+1$, which contradicts our assumption that it is undecided w.r.t. their extensions $\mathcal{K}$ and $\mathcal{L}$. $\square$

We have shown that the implication set $\mathcal{L}$ produced by the algorithm is sound and complete for $Imp(\overline{\mathcal{K}})$. Next, we show that this set is actually the Duquenne-Guigues base of $\mathbb{K}_{\overline{\mathcal{K}}}$, the formal context corresponding to the full context $\overline{\mathcal{K}}$. Since $Imp(\overline{\mathcal{K}}) = Imp(\mathbb{K}_{\overline{\mathcal{K}}})$, we call this also the *Duquenne-Guigues base of* $\overline{\mathcal{K}}$. Recall that the left-hand sides of the implications in this base are pseudo-intents of $\mathbb{K}_{\overline{\mathcal{K}}}$. Because the operator $\cdot''$ for $\mathbb{K}_{\overline{\mathcal{K}}}$ and the operator $\overline{\mathcal{K}}(\cdot)$ coincide, a subset P of M is a pseudo-intent of $\mathbb{K}_{\overline{\mathcal{K}}}$ if $P \neq \overline{\mathcal{K}}(P)$ and $\overline{\mathcal{K}}(Q) \subseteq P$ holds for all pseudo-intents $Q \subsetneq P$. We call such a set also a *pseudo-intent of* $\overline{\mathcal{K}}$.

Proposition 2.23 *The set $\mathcal{L}$ computed by Algorithm 2 is the Duquenne-Guigues base of $\overline{\mathcal{K}}$, and thus contains the minimum number of implications among all sets of implications that are sound and complete for $Imp(\overline{\mathcal{K}})$.*

Proof. From FCA we know that the Duquenne-Guigues base of a formal context, and thus also of the corresponding full context $\overline{\mathcal{K}}$, contains the minimum number of implications among all implication sets that are sound and complete for $Imp(\overline{\mathcal{K}})$. In Proposition 2.22, we have already shown that the implication set $\mathcal{L}$ produced by Algorithm 2 is sound and complete for $Imp(\overline{\mathcal{K}})$. Thus, it is enough to show that that (i) the left-hand sides L of the implications in $\mathcal{L}$ are pseudo-intents of $\overline{\mathcal{K}}$, and (ii) the corresponding right-hand sides are of the form $\overline{\mathcal{K}}(L) \setminus L$.

To show (ii), consider an implication $L \to R$ in $\mathcal{L}$. By the construction of $\mathcal{L}$, there is an index i such that $R = \mathcal{K}_i(L) \setminus L$. We know that $L \to R$ is not refuted by $\overline{\mathcal{K}}$, and thus $\mathcal{K}_i(L) = \mathcal{K}_{i+1}(L) = \ldots = \mathcal{K}(L)$ by Proposition 2.21. Thus, it is enough to show that $\mathcal{K}(L) = \overline{\mathcal{K}}(L)$. The inclusion $\overline{\mathcal{K}}(L) \subseteq \mathcal{K}(L)$ follows from the fact that $\mathcal{K} \leq \overline{\mathcal{K}}$, and the inclusion in the other direction follows from the fact that $L \to \mathcal{K}_i(L) \setminus L$, and thus also $L \to \mathcal{K}(L)$, is not refuted by $\overline{\mathcal{K}}$ (see Proposition 2.13).

To show (i), first note that the implication $L \to \mathcal{K}_i(L) \setminus L$ is only added by the algorithm to the implication set if $L \neq \mathcal{K}_i(L)$. Together with what we have shown in the proof of (ii) above, this yields $L \neq \overline{\mathcal{K}}(L)$. To show that L is indeed a pseudo-intent of $\overline{\mathcal{K}}$, we assume that Q is a pseudo-intent of $\overline{\mathcal{K}}$ such that $Q \subsetneq L$. We must show that $\overline{\mathcal{K}}(Q) \subseteq L$. By Proposition 2.13, $Q \to \overline{\mathcal{K}}(Q)$ is not refuted by $\overline{\mathcal{K}}$. Since $\mathcal{L}$ is complete for $Imp(\overline{\mathcal{K}})$ by Proposition 2.22, the implication $Q \to \overline{\mathcal{K}}(Q)$ follows from $\mathcal{L}$, i.e., $\overline{\mathcal{K}}(Q) \subseteq \mathcal{L}(Q)$. In addition, $Q \subseteq L$ implies $\mathcal{L}(Q) \subseteq \mathcal{L}(L)$, and we know from the proof of Proposition 2.22 that L is $\mathcal{L}$-closed. Thus, we have $\overline{\mathcal{K}}(Q) \subseteq \mathcal{L}(Q) \subseteq \mathcal{L}(L) = L$, which completes the proof that L is a pseudo-intent of $\overline{\mathcal{K}}$. □

In the remainder of this section, we demonstrate the execution of the algorithm on a small example.

Example 2.24 Let $M = \{m_1, m_2, m_3, m_4\}$ be a set of attributes, $\mathcal{K}_0$ an initial set of pods describing objects from some application domain, and $\overline{\mathcal{K}}$ the set of fods that represents the expert's view of this application domain. The contexts $\mathcal{K}_0$ and $\overline{\mathcal{K}}$ are shown below as crosstables, where for a pod $o_i = (A, S)$, "+" indicates that the attribute belongs to A, "−" indicates that the attribute belongs to S, and the remaining attributes are marked by "?".

$\mathcal{K}_0$	m_1	m_2	m_3	m_4
o_1	+	?	+	−
o_2	+	?	?	−
o_3	?	−	?	+

$\overline{\mathcal{K}}$	m_1	m_2	m_3	m_4
o_1	+	+	+	−
o_2	+	−	+	−
o_3	+	−	+	+
o_4	+	+	−	−

Table 2.24 shows the execution of Algorithm 2 on $\mathcal{K}_0$ and w.r.t. the underlying full context $\overline{\mathcal{K}}$.
In Step 1, the user extends the partial context by adding the new pod $(\{m_1\}, \{m_3\})$ with name o_4 as a counterexample to the implication $\varnothing \to \{m_1, m_3\}$, since this implication is refuted by $\overline{\mathcal{K}}$. In Step 2, the user accepts the implication $\varnothing \to \{m_1\}$ since it is not refuted by $\overline{\mathcal{K}}$. In addition, this new implication is used to extend the partial context. Since the implication says that every object should have attribute m_1, the entry for attribute m_1 of pod o_3 is changed to +. A similar extension is done in Step 4. After adding the new implication $\{m_1, m_4\} \to \{m_3\}$, we update the entry for attribute m_3 of pod o_3 by changing it to +. Note that

	P_i	$\mathcal{K}_i(P_i)$	refuted by $\overline{\mathcal{K}}$?	action
1	$\varnothing$	$\{m_1, m_3\}$	yes	new pod $o_4 := (\{m_1\}, \{m_3\})$
2	$\varnothing$	$\{m_1\}$	no	new imp. $\varnothing \to \{m_1\}$ set $(o_3, m_1) := +$
3	$\{m_1\}$	$\{m_1\}$		next P_i
4	$\{m_1, m_4\}$	$\{m_1, m_3, m_4\}$	no	new imp. $\{m_1, m_4\} \to \{m_3\}$ set $(o_3, m_3) := +$
5	$\{m_1, m_3\}$	$\{m_1, m_3\}$		next P_i
6	$\{m_1, m_3, m_4\}$	$\{m_1, m_3, m_4\}$		next P_i
7	$\{m_1, m_2\}$	$\{m_1, m_2, m_3, m_4\}$	yes	set $(o_1, m_2) := +$
8	$\{m_1, m_2\}$	$\{m_1, m_2, m_3\}$	yes	set $(o_4, m_2) := +$
9	$\{m_1, m_2\}$	$\{m_1, m_2\}$		next P_i
10	$\{m_1, m_2, m_3\}$	$\{m_1, m_2, m_3\}$		next P_i
11	$\{m_1, m_2, m_3, m_4\}$			

Table 1: Execution of the algorithm on $\mathcal{K}_0$ and $\overline{\mathcal{K}}$

later on, this avoids asking a redundant question to the expert. If we had not updated this value, the next P_i would also be $\{m_1, m_3\}$, but $\mathcal{K}_i(P_i)$ would be $\{m_1, m_2, m_3\}$, and the implication question "$\{m_1, m_3\} \to \{m_1, m_2, m_3\}$?" would be asked. This implication is refuted by $\overline{\mathcal{K}}$, so the user would have to provide a counterexample in order to refute it.
The execution of the algorithm continues in a similar way until in Step 11 P_i is the whole set of attributes (see the table for details). Note that, in Steps 7 and 8, instead of adding a new pod, the partial context is extended by changing existing pods, in order to turn them into counterexamples to the implication questions asked.
At the end of its execution, the algorithm has produced the following partial context $\mathcal{K}$

$\mathcal{K}$	m_1	m_2	m_3	m_4
o_1	$+$	$+$	$+$	$-$
o_2	$+$	?	?	$-$
o_3	$+$	$-$	$+$	$+$
o_4	$+$	$+$	$-$	?

and the implication bases $\{\varnothing \to \{m_1\},\ \ \{m_1, m_4\} \to \{m_3\}\}$.
Thus, at the end of the exploration, some entries in the table describing the partial context are still undetermined, i.e., marked with "?." This means that the described context is still partial and not full.

Name of constructor	Syntax	Semantics
top-concept	$\top$	$\Delta^{\mathcal{I}}$
bottom-concept	$\bot$	$\varnothing$
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
value restriction	$\forall r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y : (x,y) \in r^{\mathcal{I}} \to y \in C^{\mathcal{I}}\}$
existential restriction	$\exists r.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
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 2: Syntax and semantics of $\mathcal{ALC}$-concept descriptions, TBoxes, and ABoxes.

3 Description Logics

In the first part of this section, we briefly recall some notions regarding Description Logics (DLs). More details and references for the results mentioned below can be found in [1]. In the second part, we show how DL knowledge bases can give rise to partial contexts, and in the third part we show how attribute exploration for partial contexts can be used to complete DL knowledge bases.

3.1 Basic definitions

In order to represent knowledge about an application domain using DLs, 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 concrete individuals (objects). As an example, we consider the language $\mathcal{ALC}$, which provides for the concept constructors shown in the upper part of Table 2. In this table, r stands for a role name, C, D stand for concept descriptions, and a, b stand for individual names. An *$\mathcal{ALC}$-concept description* is either a concept name, or obtained by applying one of the concept constructors of the table to $\mathcal{ALC}$-concept descriptions. 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 2). A *knowledge base* 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 $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The semantics of arbitrary concept descriptions is defined inductively, as seen in the semantics column of Table 2. 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 also called a model of the knowledge base $(\mathcal{T}, \mathcal{A})$.

Given a TBox $\mathcal{T}$, an ABox $\mathcal{A}$, concept descriptions C, D, and an individual name a, the following are relevant *inference problems*:

- *Satisfiability:* C is satisfiable w.r.t. $\mathcal{T}$ if there is a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}} \neq \varnothing$.
- *Subsumption:* 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 $\mathcal{T}$.
- *Consistency:* the knowledge base $(\mathcal{T}, \mathcal{A})$ is consistent if it has a model.
- *Instance:* 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 $\mathcal{T}$ and $\mathcal{A}$.

For the DL $\mathcal{ALC}$ with the TBox and ABox formalisms as introduced above, the satisfiability, the subsumption, the instance, and the consistency problem are ExpTime-complete. Note that, in a DL that allows for conjunction and negation, the subsumption and the satisfiability problem are interreducable in polynomial time, and the same is true for the instance and the consistency problem. In addition, the satisfiability problem can always be reduced in polynomial time to the consistency problem. Highly optimized DL reasoners such at FaCT [10], Racer [8], and Pellet [25] can solve these problems in DLs that are considerably more expressive than $\mathcal{ALC}$.

If the TBox is empty or acyclic, then these problems are PSpace-complete. An *acyclic TBox* consists of concept definitions of the form $A \equiv C$ with A a concept name, which can be expressed by the pair of GCIs $A \sqsubseteq C, C \sqsubseteq A$. This set of concept definitions must satisfy the additional requirements that a concept name can occur at most once as a left-hand side of a definition and that there are no cyclic dependencies between the definitions.

It should be noted that the approach for completing DL knowledge bases introduced below is not restricted to $\mathcal{ALC}$. It applies to arbitary DLs, provided that some restrictions on the availability of certain constructors and on the algorithmic solvability of the above inference problems are satisfied:

- The description language must allow for conjunction and negation.

- The TBox formalism must allow for GCIs.
- The ABox formalism must allow for concept assertions.
- The subsumption, the instance, and the consistency problem must be decidable.

3.2 DLs and partial contexts

Given a consistent DL knowledge base $(\mathcal{T}, \mathcal{A})$, any individual in $\mathcal{A}$ induces a partial object description, where the set of attributes consists of concepts. To be more precise, let M be a finite set of concept descriptions. Any individual name a occurring in $\mathcal{A}$ gives rise to the partial object description

$$\begin{aligned} pod_{\mathcal{T},\mathcal{A}}(a, M) := (A, S) \ \text{ where } \ & A := \{C \in M \mid \mathcal{T}, \mathcal{A} \models C(a)\} \ \text{ and} \\ & S := \{C \in M \mid \mathcal{T}, \mathcal{A} \models \neg C(a)\}, \end{aligned}$$

and the whole ABox induces the partial context

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

Note that $pod_{\mathcal{T},\mathcal{A}}(a, M)$ is indeed a pod since $(\mathcal{T}, \mathcal{A})$ was assumed to be consistent, and thus we cannot simultaneously have $\mathcal{T}, \mathcal{A} \models C(a)$ and $\mathcal{T}, \mathcal{A} \models \neg C(a)$.

Similarly, any element $d \in \Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ gives rise to the full example

$$\begin{aligned} fod_{\mathcal{I}}(d, M) := (\overline{A}, \overline{S}) \ \text{ where } \ & \overline{A} := \{C \in M \mid d \in C^{\mathcal{I}}\} \ \text{ and} \\ & \overline{S} := \{C \in M \mid d \in (\neg C)^{\mathcal{I}}\}, \end{aligned}$$

and the whole interpretation induces the full context

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

Note that $fod_{\mathcal{I}}(d, M)$ is indeed a fod since every $d \in \Delta^{\mathcal{I}}$ satisfies either $d \in C^{\mathcal{I}}$ or $d \in \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} = (\neg C)^{\mathcal{I}}$.

Proposition 3.1 *Let $(\mathcal{T}, \mathcal{A})$ be a consistent knowledge base, M a set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}, \mathcal{A})$. Then $\mathcal{K}_{\mathcal{I}}(M)$ is a realizer of $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$.*

Proof. Consider a pod $(A, S) \in \mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$, i.e., $(A, S) = pod_{\mathcal{T},\mathcal{A}}(a, M)$, where a is an individual name occurring in $\mathcal{A}$. We claim that (A, S) is realized by $(\overline{A}, \overline{S}) := fod_{\mathcal{I}}(a^{\mathcal{I}}, M) \in \mathcal{K}_{\mathcal{I}}(M)$.

Let C be an element of A, i.e., $\mathcal{T}, \mathcal{A} \models C(a)$. Since $\mathcal{I}$ is a model of $(\mathcal{T}, \mathcal{A})$, this implies $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and thus $C \in \overline{A}$. This shows $A \subseteq \overline{A}$. The inclusion $S \subseteq \overline{S}$ can be shown accordingly. $\square$

The notion of refutation of an implication is transferred from partial (full) contexts to knowledge bases (interpretations) in the obvious way.

Definition 3.2 *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 *$\mathcal{I}$. The set of implications over M that hold in $\mathcal{I}$ is denoted by $Imp_M(\mathcal{I})$. 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, $L \rightarrow R$ is refuted by $(\mathcal{T}, \mathcal{A})$ iff there is an individual name a occurring in $\mathcal{A}$ such that $\mathcal{T}, \mathcal{A} \models C(a)$ for all $C \in L$ and $\mathcal{T}, \mathcal{A} \models \neg D(a)$ for some $D \in R$. Similarly, $L \rightarrow R$ is refuted by $\mathcal{I}$ iff there is an element $d \in \Delta^{\mathcal{I}}$ such that $d \in C^{\mathcal{I}}$ for all $C \in L$ and $d \notin D^{\mathcal{I}}$ for some $D \in R$. In addition, the implication $L \rightarrow R$ holds in $\mathcal{I}$ iff $(\sqcap L)^{\mathcal{I}} \subseteq (\sqcap R)^{\mathcal{I}}$.

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

Proof. If $L \rightarrow R$ follows from $\mathcal{T}$, then $(\sqcap L)^{\mathcal{I}} \subseteq (\sqcap R)^{\mathcal{I}}$ holds since $\mathcal{I}$ is a model of $\mathcal{T}$. This shows that $L \rightarrow R$ holds in $\mathcal{I}$. $\square$

The operator $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)(\cdot)$ induced by the partial context $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$ is defined as in Proposition 2.13. Since in the following the attribute set M can be assumed to be fixed, we will write $\mathcal{K}_{\mathcal{T},\mathcal{A}}$ rather that $\mathcal{K}_{\mathcal{T},\mathcal{A}}(M)$. Obviously, the result of applying this operator to a set $P \subseteq M$ can be described as follows:

$$\mathcal{K}_{\mathcal{T},\mathcal{A}}(P) = M \setminus \bigcup \{D \in M \mid \exists a.\ P \subseteq \{C \mid \mathcal{T}, \mathcal{A} \models C(a)\} \wedge \mathcal{T}, \mathcal{A} \models \neg D(a)\}$$

By Proposition 2.13, $\mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is the largest subset of M such that $P \rightarrow \mathcal{K}_{\mathcal{T},\mathcal{A}}(P)$ is not refuted by $(\mathcal{T}, \mathcal{A})$.

3.3 Completion of DL knowledge bases

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 in the previous subsection, this can formally be defined as follows.

Definition 3.4 *Let $(\mathcal{T}, \mathcal{A})$ be a 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 simply* 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 that also has $\mathcal{I}$ as a model. 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 $\mathcal{A}_0 \subseteq \mathcal{A}$.*

In order to rephrase the definition of completeness, let us say that the element $d \in \Delta^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ *satisfies* the subsumption statement $C \sqsubseteq D$ if $d \notin C^{\mathcal{I}}$ or $d \in D^{\mathcal{I}}$, and that $\mathcal{I}$ *satisfies* this statement if every element of $\Delta^{\mathcal{I}}$ satisfies it. In addition, let us call the individual name a a *counterexample in* $(\mathcal{T}, \mathcal{A})$ to the subsumption statetment $C \sqsubseteq D$ if $\mathcal{T}, \mathcal{A} \models C(a)$ and $\mathcal{T}, \mathcal{A} \models \neg D(a)$.

Lemma 3.5 *The knowledge base $(\mathcal{T}, \mathcal{A})$ is complete w.r.t. its model $\mathcal{I}$ iff the following statements are equivalent for all subsets L, R of M:*

1. *$\sqcap L \sqsubseteq \sqcap R$ is satisfied by $\mathcal{I}$;*
2. *$\sqcap L \sqsubseteq_{\mathcal{T}} \sqcap R$ holds;*
3. *$(\mathcal{T}, \mathcal{A})$ does not contain a counterexample to $\sqcap L \sqsubseteq \sqcap R$.*

In the following, we use an adaptation of the attribute exploration algorithm for partial contexts presented in the previous section in order 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 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 $L \to R$ is 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 must extend the current ABox (by adding assertions) 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

Algorithm 3 Completion of DL knowledge bases

Input: $\mathcal{T}_0$, $\mathcal{A}_0$, M $\{(\mathcal{T}_0, \mathcal{A}_0)$ has the underlying interpretation $\mathcal{I}$ as model$\}$
$i := 0$
$\mathcal{L}_0 := \varnothing$ {initial empty set of implications}
$P_0 := \varnothing$ {lectically smallest $\mathcal{L}_0$-closed subset of M}
while $P_i \neq M$ **do**
  Compute $\mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$
  if $P_i \neq \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ **then** {check whether the implication follows from $\mathcal{T}_i$}
    if $\sqcap P_i \sqsubseteq_{\mathcal{T}_i} \sqcap \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ **then**
      $\mathcal{A}_{i+1} := \mathcal{A}_i$
      $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \to \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i) \setminus P_i\}$
      $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
    else
      Ask the expert if $P_i \to \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is refuted by $\mathcal{I}$.
      if no **then** $\{\sqcap P_i \sqsubseteq \sqcap \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is satisfied in $\mathcal{I}\}$
        $\mathcal{A}_{i+1} := \mathcal{A}_i$
        $\mathcal{L}_{i+1} := \mathcal{L}_i \cup \{P_i \to \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i) \setminus P_i\}$
        $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
        $\mathcal{T}_{i+1} := \mathcal{T}_i \cup \{\sqcap P_i \sqsubseteq \sqcap(\mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i) \setminus P_i)\}$ {extend the TBox}
      else
        Get an ABox $\mathcal{A}'$ from the expert such that $\mathcal{A}_i \subseteq \mathcal{A}'$, $\mathcal{I}$ is a model of $\mathcal{A}'$, and $P_i \to \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is refuted by $\mathcal{A}'$
        $\mathcal{A}_{i+1} := \mathcal{A}'$ {extend the ABox}
        $\mathcal{T}_{i+1} = \mathcal{T}_i$
        $P_{i+1} := P_i$
        $\mathcal{L}_{i+1} := \mathcal{L}_i$
      end if
    end if
  else
    $\mathcal{A}_{i+1} := \mathcal{A}_i$
    $\mathcal{T}_{i+1} := \mathcal{T}_i$
    $\mathcal{L}_{i+1} := \mathcal{L}_i$
    $P_{i+1} := \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$ for the max. j that satisfies $P <_j \mathcal{L}_{i+1}((P_i \cap \{m_1, \ldots, m_{j-1}\}) \cup \{m_j\})$
  end if
  $i := i + 1$
end while

$L \to R$ cannot be refuted by $\mathcal{I}$. This completion algorithm for DL knowledge bases is described in more detail in Algorithm 3.

Note that Algorithm 3 applied to $\mathcal{T}_0, \mathcal{A}_0, M$ with the underlying model $\mathcal{I}$ of $(\mathcal{T}_0, \mathcal{A}_0)$ behaves identical to Algorithm 2 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 is an immediate consequence of the following facts:

1. for all $i \geq 0$, the underlying interpretation $\mathcal{I}$ is a model of $(\mathcal{T}_i, \mathcal{A}_i)$;

2. if the test $\sqcap P_i \sqsubseteq_{\mathcal{T}_i} \sqcap \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is successful, then the implication $P_i \rightarrow \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ holds in $\mathcal{I}$, and thus the expert would have answered "no" to this implication question;

3. if $\mathcal{T}'$ is a TBox such that $\mathcal{T}_i \subseteq \mathcal{T}'$ and $\mathcal{I}$ is a model of $\mathcal{T}'$, then $\mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(M) \leq \mathcal{K}_{\mathcal{T}',\mathcal{A}_i}(M) \leq \mathcal{K}_{\mathcal{I}}(M)$;

4. if $\mathcal{A}'$ is an ABox such that $\mathcal{A}_i \subseteq \mathcal{A}'$, $\mathcal{I}$ is a model of $\mathcal{A}'$, and $P_i \rightarrow \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is refuted by $\mathcal{A}'$, then $\mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(M) \leq \mathcal{K}_{\mathcal{T}_i,\mathcal{A}'}(M) \leq \mathcal{K}_{\mathcal{I}}(M)$ and $P_i \rightarrow \mathcal{K}_{\mathcal{T}_i,\mathcal{A}_i}(P_i)$ is refuted by $\mathcal{K}_{\mathcal{T}_i,\mathcal{A}'}(M)$.

Thus, Proposition 2.22 immediately implies the following proposition.

Proposition 3.6 *Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a knowledge base, M a finite set of concept descriptions, and $\mathcal{I}$ a model of $(\mathcal{T}_0, \mathcal{A}_0)$. Then Algorithm 3 terminates, and upon termination outputs a knowledge base $(\mathcal{T}, \mathcal{A})$ and a set of implications $\mathcal{L}$ such that*

- *$\mathcal{L}$ is sound and complete for $Imp_M(\mathcal{I})$, and*
- *$(\mathcal{T}, \mathcal{A})$ refutes every implication that is refuted by $\mathcal{I}$.*

It remains to show that Algorithm 3 really computes a completion of the input knowledge base.

Theorem 3.7 *Let $(\mathcal{T}_0, \mathcal{A}_0)$ be a 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 3. Then $(\mathcal{T}, \mathcal{A})$ is a completion of $(\mathcal{T}_0, \mathcal{A}_0)$.*

Proof. Obviously, $(\mathcal{T}, \mathcal{A})$ extends $(\mathcal{T}_0, \mathcal{A}_0)$ and has $\mathcal{I}$ as a model. To prove that $(\mathcal{T}, \mathcal{A})$ is complete, we must show that 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. $L \to R$ is not refuted by $(\mathcal{T}, \mathcal{A})$.

"$2 \to 1$" is an immeditate consequence of the fact that $\mathcal{I}$ is a model of $\mathcal{T}$, and "$1 \to 2$" follows from the facts that $\mathcal{L}$ is complete for $Imp_M(\mathcal{I})$ and $\mathcal{T}$ contains the GCIs $\sqcap L' \sqsubseteq \sqcap R'$ for all implications $L' \to R'$ in $\mathcal{L}$.

"$1 \to 3$" is an immediate consequence of the fact that $\mathcal{I}$ is a model of $\mathcal{A}$, and "$3 \to 1$" of the fact that $(\mathcal{T}, \mathcal{A})$ refutes every implication that is refuted by $\mathcal{I}$. □

4 Implementation

Based on the ideas and 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 so-called Instance Store (iS) [11]. The iS is based on a hybrid architecture that combines a DL reasoner and a relational database in order to support efficient reasoning with a very large number of individuals. Our reason for using the iS tool was that our completion approach works better (i.e., asks less questions to the expert) if the ABox already contains a large number of individuals. In fact, the more individuals are already present, the less new ones need to be created by the expert as counterexamples to implication questions. The iS can only deal with role-free ABoxes, i.e., ABoxes that do not contain role assertions, but this is sufficient for the human protein phosphatases ontology mentioned as a motivating example in the introduction.

A more problematic restriction is the fact that the iS does not support incremental reasoning when the knowledge base is extended. Extending the TBox is not supported at all by its user interface, and though adding new concept assertions is possible, the frequent extensions required by our Algorithm 3 are computationally quite expensive. For this reason, our first implementation does not continuously extend the underlying DL knowledge base during the run of the exploration algorithm, but only once when the exploration process is finished.

To be more precise, the completion tool works as follows. First, it asks the expert for a concept description C_0 describing the part of the knowledge base that is to be explored. This concept description determines the set of relevant attributes:

$$M := \{A \mid A \text{ is a concept name occurring in } (\mathcal{T}_0, \mathcal{A}_0) \text{ such that } A \sqsubseteq_{\mathcal{T}_0} C_0\}.$$

Based on M and the knowledge base $(\mathcal{T}_0, \mathcal{A}_0)$, the initial partial context $\mathcal{K}_{\mathcal{T}_0,\mathcal{A}_0}(M)$ is computed. In principle, the current version of our tool then uses Algorithm 2 to explore this partial context. However, before an implication question is given to the expert, the tool first checks whether the corresponding subsumption relationship follows from the TBox. If this is the case, the implication is accepted without asking the expert. Whenever an implication is added, the current context $\mathcal{K}_i$ is extended to $\mathcal{L}(\mathcal{K}_i)$. In the case where an implication does not hold in the full context induced by the underlying model of the knowledge base, the expert is offered the opportunity to extend the partial context by either adding a new pod as counterexample or by extending an already existing pod. The system supports the extension of existing pods by showing to the user all pods from the current context that have an extension to a counterexample of the implication in question. Before adding a new pod (A, S) or extending an existing one to (A, S), the system checks whether this pod really is a counterexample to the implication question and whether it is consistent with the TBox, i.e., whether $\sqcap A \sqcap \bigsqcap_{C \in S} \neg C$ is satisfiable w.r.t. $\mathcal{T}_0$.[2] When the attribute exploration is finished with a final partial context $\mathcal{K}$ and a finite set of implications $\mathcal{L}$, the TBox $\mathcal{T}_0$ is extended to

$$\mathcal{T} := \mathcal{T}_0 \cup \{\sqcap L \sqsubseteq \sqcap R \mid L \to R \in \mathcal{L}\}$$

and the ABox $\mathcal{A}_0$ is extended to

$$\mathcal{A} := \mathcal{A}_0 \cup \{C(a_{(A,S)}), \neg D(a_{(A,S)}) \mid (A, S) \in \mathcal{K} \setminus \mathcal{K}_{\mathcal{T}_0,\mathcal{A}_0}(M), C \in A, D \in S\},$$

where $a_{(A,S)}$ is a new individual name created for the pod (A, S).[3] It is easy to show that the statement of Theorem 3.7 also holds in this case, i.e., the knowledge base $(\mathcal{T}, \mathcal{A})$ computed by our current implementation is a completion of the input knowledge base $(\mathcal{T}_0, \mathcal{A}_0)$.

Viewed in the light of Algorithm 2, of which Algorithm 3 is an instance, the main difference between our current implementation and Algorithm 3 is the following. Since the TBox and the ABox are not extended during the exploration process, the extension $\mathcal{K}'$ used in lines 11 and 16 of Algorithm 2 may be smaller than the ones that would have been created by Algorithm 3. This does not compromise correctness of the algorithm, but it may result in more implication questions being asked to the expert.

5 Conclusion

We have described a knowledge acquisition method that allows to extend a Description Logic knowledge base $(\mathcal{T}_0, \mathcal{A}_0)$ by additional information on the relationships that hold in a specific interpretation $\mathcal{I}$ between concepts in a set of concept descriptions M deemed to be interesting. The method extends the TBox of the knowledge base by additional GCIs for relationships that hold in $\mathcal{I}$, but do not follow from the TBox; and it extends the ABox by counterexamples to relationships that do not hold in $\mathcal{I}$, and are not yet refuted by the existing individuals in

[2]This allows the system to detect errors made by the expert. In the formulation of our algorithms, we have assumed that the expert only gives answers that are correct w.r.t. the underlying full context (model of the knowledge base). However, in practice, errors probably cannot be avoided, and thus the system should at least try to detect “obvious” ones.

[3]Usually, this name will be provided by the expert.

the ABox. To obtain the necessary information on whether a relationship holds or not, the existing TBox and ABox are checked first. Only if they cannot be used to decide this question, an expert that "knows" $\mathcal{I}$ is asked. The method is based on the well-known attribute exploration approach from Formal Concept Analysis. However, this approach had to be extended in order to be able to handle partial contexts, to correctly represent the open-world semantics of DL knowledge bases.

As a formalization of what "relationships between the concepts in M" really means, we have used subsumption relationships between arbitrary conjunctions of elements of M. The reason was, on the one hand, that these relationships should be fairly easy to decide by a domain expert. On the other hand, the close connection between such relationships and implications, as considered in Formal Concept Analysis, facilitated the adaptation of attribute exploration for our purposes. One could also be interested in more complex relationships, however. For example, one could fix a specific description language $\mathcal{D}$ (e.g., comprising some subset of the constructors of $\mathcal{ALC}$), then take as attributes $\mathcal{D}$-concept descriptions over the concept "names" from M, and ask for all subsumption relationships between the conjunctions of the concept descriptions obtained this way. The immediate disadvantage of this extension is that in general the set of attributes is no longer finite, 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 [21]. 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 the method described in [21] to our purposes not only requires an extension of this method to partial contexts, but also some new ideas of how to deal with the practicality issue.

Regarding our implementation, we intend to migrate it from the Instance Store as underlying reasoner to the system Pellet [25]. Pellet can also reason efficiently with a large number of individuals, but in contrast to the Instance Store, it supports role assertion and incremental reasoning [19, 9]. Thus, using Pellet will allow us to implement Algorithm 3, and compare its performance (in particular w.r.t. the number of implication question asked) with the one of our current implementation.

References

[1] 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.

[2] S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: A reason-able ontology editor for the semantic web. In F. Baader, G. Brewka, and T. Eiter, editors, *KI-2001: Advances in Artificial Intelligence*, volume 2174 of *Lecture Notes in Artificial Intelligence*, pages 396–408. Springer-Verlag, 2001.

[3] P. Burmeister and R. Holzer. On the treatment of incomplete knowledge in formal concept analysis. In *Proceedings of the 8th International Conference on Conceptual Structures, (ICCS 2000)*, volume 1867 of *Lecture Notes in Computer Science*, pages 385–398. Springer-Verlag, 2000.

[4] P. Burmeister and R. Holzer. Treating incomplete knowledge in formal concept analysis. In *Formal Concept Analysis*, volume 3626 of *Lecture Notes in Computer Science*, pages 114–126. Springer-Verlag, 2005.

[5] W. F. Dowling and J. H. Gallier. Linear-time algorithms for testing the satisfiability of propositional horn formulae. *Journal of Logic Programming*, 1(3):267–284, 1984.

[6] B. Ganter. Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany, 1984.

[7] B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer-Verlag, Berlin, Germany, 1999.

[8] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[9] C. Halaschek-Wiener, B. Parsia, E. Sirin, and A. Kalyanpur. Description logic reasoning for dynamic ABoxes. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, CEUR Electronic Workshop Proceedings, pages 200–207, 2006.

[10] I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[11] I. Horrocks, L. Li, D. Turi, and S. Bechhofer. The instance store: DL reasoning with large numbers of individuals. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 31–40, 2004.

[12] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From SHIQ and RDF to OWL: The making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.

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

[14] A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in owl ontologies. In *Proc. of the 3rd Eur. Semantic Web Conference (ESWC'06)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2006.

[15] 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 *Proceedings of the Third International Semantic Web Conference*, Hiroshima, Japan, 2004.

[16] D. Maier. *The Theory of Relational Databases.* Computer Science Press, Maryland, 1983.

[17] D. Oberle, R. Volz, B. Motik, and S. Staab. An extensible ontology software environment. In S. Staab and R. Studer, editors, *Handbook on Ontologies*, International Handbooks on Information Systems, pages 311–333. Springer-Verlag, 2004.

[18] S. A. Obiedkov. Modal logic for evaluating formulas in incomplete contexts. In *Proceedings of the 10th International Conference on Conceptual Structures, (ICCS 2002)*, volume 2393 of *Lecture Notes in Computer Science*, pages 314–325. Springer-Verlag, 2002.

[19] B. Parsia, C. Halaschek-Wiener, and E. Sirin. Towards incremental reasoning through updates in OWL-DL. In *Reasoning on the Web (RoW2006), Workshop at WWW2006*, 2006.

[20] B. Parsia, E. Sirin, and A. Kalyanpur. Debugging OWL ontologies. In A. Ellis and T. Hagino, editors, *Proc. of the 14th International Conference on World Wide Web (WWW'05)*, pages 633–640. ACM, 2005.

[21] S. Rudolph. Exploring relational structures via $\mathcal{FLE}$. In K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, editors, *12th International Conference on Conceptual Structures (ICCS 2004)*, volume 3127 of *Lecture Notes in Computer Science*, pages 196–212, 2004.

[22] S. Schlobach. Debugging and semantic clarification by pinpointing. In A. Gómez-Pérez and J. Euzenat, editors, *Proc. of the 2nd Eur. Semantic Web Conference (ESWC'05)*, volume 3532 of *Lecture Notes in Computer Science*, pages 226–240. Springer-Verlag, 2005.

[23] S. Schlobach. Diagnosing terminologies. In M. M. Veloso and S. Kambhampati, editors, *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 670–675. AAAI Press/The MIT Press, 2005.

[24] S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors,

Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003), pages 355–362, Acapulco, Mexico, 2003. Morgan Kaufmann, Los Altos.

[25] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, pages 212–213, 2004.

[26] R. Wille. Restructuring lattice theory: An approach based on hierarchies of concepts. In I. Rival, editor, *Ordered Sets*, pages 445–470. Reidel, Dordrecht-Boston, 1982.

[27] 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 the 4th Int. Semantic Web Conf. (ISWC'05)*, volume 3729 of *Lecture Notes in Computer Science*, pages 786–800. Springer-Verlag, 2005.