A Survey on how Description Logic Ontologies Benefit from FCA

Barış Sertkaya

SAP Research Center Dresden, Germany
`baris.sertkaya@sap.com`

Abstract. Although the notion of a concept as a collection of objects sharing certain properties, and the notion of a conceptual hierarchy are fundamental to both Formal Concept Analysis and Description Logics, the ways concepts are described and obtained differ significantly between these two research areas. Despite these differences, there have been several attempts to bridge the gap between these two formalisms, and attempts to apply methods from one field in the other. The present work aims to give an overview on the research done in combining Description Logics and Formal Concept Analysis.

1 Introduction

Formal Concept Analysis (FCA) [28] is a field of applied mathematics that aims to formalize the notions of a concept and a conceptual hierarchy by means of mathematical tools. On the other hand Description Logics (DLs) [3] are a class of logic-based knowledge representation formalisms that are used to represent the conceptual knowledge of an application domain in a structured way. Although the notion of a concept as a collection of objects sharing certain properties, and the notion of a conceptual hierarchy are fundamental to both FCA and DLs, the ways concepts are described and obtained differ significantly between these two research areas. In DLs, the relevant concepts of the application domain are formalized by so-called concept descriptions, which are expressions built from unary predicates (that are called atomic concepts), and binary predicates (that are called atomic roles) with the help of the concept constructors provided by the DL language. Then in a second step, these concept descriptions are used to describe properties of individuals occurring in the domain, and the roles are used to describe relations between these individuals. On the other hand, in FCA, one starts with a so-called formal context, which in its simplest form is a way of specifying which attributes are satisfied by which objects. A formal concept of such a context is a pair consisting of a set of objects called extent, and a set of attributes called intent such that the intent consists of exactly those attributes that the objects in the extent have in common, and the extent consists of exactly those objects that share all attributes in the intent.

There are several differences between these approaches. First, in FCA one starts with a purely extensional description of the application domain, and then

derives the formal concepts of this specific domain, which provide a useful structuring. In a way, in FCA the intensional knowledge is obtained from the extensional part of the knowledge. On the other hand, in DLs the intensional definition of a concept is given independently of a specific domain (interpretation), and the description of the individuals is only partial. Second, in FCA the properties are atomic, and the intensional description of a formal concept (by its intent) is just a conjunction of such properties. DLs usually provide a richer language for the intensional definition of concepts, which can be seen as an expressive, yet decidable sublanguage of first-order predicate logic.

Despite these differences, there have been several attempts to bridge the gap between these two formalisms, and attempts to apply methods from one field to the other. For example, there have been efforts to enrich FCA with more complex properties similar to concept constructors in DLs [60, 45, 44, 23, 46]. On the other hand, DL research has benefited from FCA methods to solve some problems encountered in knowledge representation using DLs [1, 55, 10, 12, 48, 14, 49, 52, 16, 7, 53, 50, 4, 5, 13]. The present work aims to give an overview on these works done for bridging the gap between the two formalisms. In Section 2 we give a short introduction to DLs without going into technical details. We assume that the reader is familiar with FCA. We do not introduce FCA, we refer the reader to [28] for details. In Section 3 we summarize the existing work done by other researchers in the field. In Section 4 we summarize our own contributions to the field, and conclude with Section 5.

2   Description Logics

Description Logics (DLs) [3] are a class of knowledge representation formalisms that are used to represent the terminological knowledge of an application domain in a structured way. Since their introduction, DLs have been used in various application domains such as medical informatics, software engineering, configuration of technical systems, natural language processing, databases and web-based information systems. But their most notable success so far is the adoption of the DL-based language OWL[1][34] as the standard ontology language for the semantic web [17].

Syntax In DLs, one formalizes the relevant notions of an application domain by *concept descriptions*. A concept description is an expression built from atomic concepts, which are unary predicates, and atomic roles, which are binary predicates, by using the concept constructors provided by the particular DL language in use. DL languages are identified with the concept constructors they allow. For instance the smallest propositionally closed language allowing for the constructors $\sqcap$ (conjunction), $\sqcup$ (disjunction), $\neg$ (negation), $\forall$ (value restriction) and $\exists$ (existential restriction) is called $\mathcal{ALC}$.

[1] Web Ontology Language. See `http://www.w3.org/TR/owl-features`

Typically, a *DL knowledge base* consists of a *terminological box (TBox)*, which defines the terminology of an application domain, and an *assertional box (ABox)*, which contains facts about a specific world. In its simplest form, a TBox is a set of *concept definitions* of the form $A \equiv C$ that assigns the concept name A to the concept description C. We call a finite set of *general concept inclusion (GCI)* axioms a *general TBox*. A GCI is an expression of the form $C \sqsubseteq D$, where C and D are two possibly complex concept descriptions. It states a subconcept/superconcept relationship between the two concept descriptions. An ABox is a set of *concept assertions* of the form $C(a)$, which means that the individual a is an instance of the concept C, and *role assertions* of the form $R(a, b)$, which means that the individual a is in R-relation with individual b.

For instance the following TBox contains the definition of a landlocked country, which is a country that only has borders on land, and the definition of an ocean country that has a border to an ocean.

$$\mathcal{T} := \{\mathsf{LandlockedCountry} \equiv \mathsf{Country} \sqcap \forall \mathsf{hasBorderTo.Land}$$
$$\mathsf{OceanCountry} \equiv \mathsf{Country} \sqcap \exists \mathsf{hasBorderTo.Ocean}\}$$

The following ABox states the facts about the individuals *Portugal*, *Austria*, and *Atlantic Ocean*.

$$\mathcal{A} := \{\mathsf{LandlockedCountry}(Austria), \mathsf{Country}(Portugal), \mathsf{Ocean}(Atlantic\ Ocean),$$
$$\mathsf{hasBorderTo}(Portugal, Atlantic\ Ocean)\}$$

Semantics The meaning of DL concepts is given by means of an *interpretation* $\mathcal{I}$, which is a tuple consisting of a *domain* $\Delta^{\mathcal{I}}$ and an *interpretation function* $\cdot^{\mathcal{I}}$. The interpretation function maps every concept occurring in the TBox to a subset of the domain, every role to a binary relation on the domain, and every individual name occurring in the ABox to an element of the domain. The meaning of complex concept descriptions is given inductively based on the constructors used in the concept description.

For instance, the concept description $\mathsf{Country} \sqcap \exists \mathsf{hasBorderTo.Ocean}$ is interpreted as the intersection of the set of countries and the set of elements of the domain that have a border to an ocean. We say that an interpretation $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if it satisfies all concept definitions in $\mathcal{T}$, i.e., for every concept definition $A \equiv C$ in $\mathcal{T}$, it maps A and C to the same subset of the domain. Similarly, we say that $\mathcal{I}$ is a model of an ABox $\mathcal{A}$, if it satisfies all concept and role assertions in $\mathcal{A}$, i.e., for every concept assertion $A(a)$ in $\mathcal{A}$, the interpretation of a is an element of the interpretation of A, and for every role assertion $r(a, b)$ the interpretation of r contains the pair consisting of the interpretations of a and b. The semantics of DL ABoxes is the *open-world semantics*, i.e., absence of information about an individual is not interpreted as negative information, but it only indicates lack of knowledge about that individual.

Inferences In an application, once we get a description of the application domain using DLs as described above, we can make inferences, i.e., deduce implicit

consequences from the explicitly represented knowledge. The basic inference on concept descriptions is *subsumption*. Given two concept descriptions C and D, the subsumption problem $C \sqsubseteq D$ is the problem of checking whether the concept description D is more general than the concept description C. In other words, it is the problem of determining whether the first concept always, i.e., in every interpretation denotes a subset of the set denoted by the second one. We say that C is subsumed by D w.r.t. a TBox $\mathcal{T}$, if in every model of $\mathcal{T}$, D is more general than C, i.e., the interpretation of C is a subset of the interpretation of D. We denote this as $C \sqsubseteq_{\mathcal{T}} D$. For instance, in the example above, the concepts LandlockedCountry and OceanCountry are both trivially subsumed by the concept Country.

The typical inference problem for ABoxes is *instance checking*, which is the problem of deciding whether the interpretation of a given individual is an element of the interpretation of a given concept in every common model of the TBox and the ABox. For instance, from $\mathcal{T}$ and $\mathcal{A}$ given above it follows that *Portugal* is an ocean country, although $\mathcal{A}$ does not contain the assertion OceanCountry(*Portugal*). Modern *DL systems* like FaCT++ [57], Racer [29], Pellet [54], KAON2 [40], Hermit [41] and CEL [9] provide their users with inference services that solve these inference problems, which are also known as *standard inferences*.

3 Existing work on DLs and FCA

The existing work done by other researchers towards bridging the gap between FCA und DLs, and attempts to apply methods from one field to the other can roughly be collected under two categories:

- efforts to enrich the language of FCA by borrowing constructors from DL languages [60, 45, 44, 23, 46]
- efforts to employ FCA methods in the solution of problems encountered in knowledge representation with DLs [1, 55, 10, 48–50, 4, 21, 20, 5]

Below we are going to discuss some of these efforts briefly.

3.1 Enriching FCA with DL constructors

Theory-driven logical scaling In [45], Prediger and Stumme have used DLs in *Conceptual Information Systems*, which are data analysis tools based on FCA. They can be used to extract data from a relational database and to store it in a formal context by using so-called *conceptual scales*. Prediger and Stumme have combined DLs with attribute exploration in order to define a new kind of conceptual scale. In this approach, DLs provide a rich language to specify which FCA attributes cannot occur together, and a DL reasoner is used during the attribute exploration process as an expert to answer the implication questions, and to provide a counterexample whenever the implication does not hold.

Terminological attribute logic In [44], Prediger has worked on introducing logical constructors into FCA. She has enriched FCA with relations, existential and universal quantifiers, and negation, obtaining a language like the DL $\mathcal{ALC}$, which she has called *terminologische Merkmalslogik* (*terminological attribute logic*[2]). In the same work she has also presented applications of her approach in enriching formal contexts with new knowledge, applications in many valued formal contexts, and applications for so-called *scales*, which are formal contexts that are used to obtain a standard formal context from a many valued formal context.

Relational concept analysis In [46], Rouane et al. have presented a combination of FCA and DLs that is called *relational concept analysis*. It is an adaptation of FCA that is intended for analyzing objects described by relational attributes in data mining. The approach is based on a collection of formal contexts called *relational context family* and relations between these contexts. The relations between the contexts are binary relations between pairs of object sets that belong to two different contexts. Processing these contexts and relations with relational concept analysis methods yields a set of concept lattices (one for each input context) such that the formal concepts in different lattices are linked by relational attributes, which are similar to roles in DLs, or associations in UML. One distinguishing feature of this approach from the other efforts that introduce relations into FCA is that the formal concepts and relations between formal concepts of different contexts can be mapped into concept descriptions in a sublanguage of $\mathcal{ALE}$, which is called $\mathcal{FL}^-\mathcal{E}$ in [46]. $\mathcal{FL}^-\mathcal{E}$ allows for conjunction, value restriction, existential restriction, and top and bottom concepts. In this approach, after the formal concepts and relations have been obtained and mapped into $\mathcal{FL}^-\mathcal{E}$ concept descriptions, DL reasoning is used to classify and check the consistency of these descriptions.

3.2   Applying FCA methods in DLs

Subsumption hierarchy of conjunctions of DL concepts In [1], Baader has used FCA for an efficient computation of an extended subsumption hierarchy of a set of DL concepts. More precisely, he used attribute exploration for computing the subsumption hierarchy of all conjunctions of a set of DL concepts. The main motivation for this work was to determine the interaction between defined concepts, which might not easily be seen by just looking at the subsumption hierarchy of defined concepts. In order to explain this, the following example has been given: assume that the defined concept NoDaughter stands for those people who have no daughters, NoSon stands for those people who have no sons, and NoSmallChild stands for those people who have no small children. Obviously, there is no subsumption relationship between these three concepts. On the other hand, the conjunction NoDaughter ⊓ NoSon is subsumed by NoSmallChild, i.e.,

---

[2] This translation is ours.

if an individual a belongs to NoSon and NoDaughter, it also belongs to NoSmallChild. However, this cannot be derived from the information that a belongs to NoSon and NoDaughter by just looking at the subsumption hierarchy. This small example demonstrates that runtime inferences concerning individuals can be made faster by precomputing the subsumption hierarchy not only for defined concepts, but also for all conjunctions of defined concepts.

To this purpose, Baader defined a formal context whose attributes were the defined DL concepts, and whose objects were all possible counterexamples to subsumption relationships, i.e., interpretations together with an element of the interpretation domain. This formal context has the property that its concept lattice is isomorphic to the required subsumption hierarchy, namely the subsumption hierarchy of conjunctions of the defined DL concepts. However, this formal context has the disadvantage that a standard subsumption algorithm can not be used as expert for this context within attribute exploration. In order to overcome this problem, the approach was reconsidered in [12] and a new formal context that has the same properties but for which a usual subsumption algorithm could be used as expert was introduced.

Subsumption hierarchy of conjunctions and disjunctions of DL concepts In [55], Stumme has extended the abovementioned subsumption hierarchy further with disjunctions of DL concepts. More precisely, he presented how the complete lattice of all possible combinations of conjunctions and disjunctions of the concepts in a DL TBox can be computed by using FCA. To this aim, he used another knowledge acquisition tool of FCA instead of attribute exploration, namely *distributive concept exploration* [56]. In the lattice computed by this method, the supremum of two DL concepts in the lattice corresponds to the disjunction of these concepts.

Subsumption hierarchy of least common subsumers In [10] Baader and Molitor have used FCA for supporting bottom-up construction of DL knowledge bases. In the bottom-up approach, the knowledge engineer does not directly define the concepts of her application domain, but she gives typical examples of a concept, and the system comes up with a concept description for these examples. The process of computing such a concept description consists of first computing the *most specific concepts* that the given examples belong to, and then computing the *least common subsumer* of these concepts. Here the choice of examples is crucial for the quality of the resulting concept description. If the examples are too similar, the resulting concept description will be too specific; conversely, if they are too distinct, the resulting concept description will be too general. In order to overcome this, Baader and Molitor have used attribute exploration for computing the subsumption hierarchy of all least common subsumers of a given set of concepts. In this hierarchy one can easily see the position of the least concept description that the chosen examples belong to, and decide whether these examples are appropriate for obtaining the intended concept description. However, there may be exponentially many least common subsumers, and depending

on the DL in use, both the least common subsumer computation and subsumption test can be expensive operations. The use of attribute exploration provides us with complete information on how this hierarchy looks like without explicitly computing all least common subsumers and classifying them.

Relational exploration In his Ph.D thesis [49], Rudolph has combined DLs and FCA for acquiring complete relational knowledge about an application domain. In his approach, which he calls *relational exploration*, he uses DLs for defining FCA attributes, and FCA for refining DL knowledge bases. More precisely, DLs makes use of the interactive knowledge acquisition method of FCA, and FCA benefits from DLs in terms of expressing relational knowledge.

In [48, 49], Rudolph uses the DL $\mathcal{FLE}$ for this purpose, which is the DL that allows for the constructors conjunction, existential restriction, and value restriction. In his previous work [47], he uses the DL $\mathcal{EL}$, which allows for the constructors conjunction and existential restriction. In both cases, he defines the semantics by means of a special pair of formal contexts called *binary power context family*, which are used for expressing relations in FCA. Binary power context families have also been used for giving semantics to *conceptual graphs*. In order to collect information about the formulae expressible in $\mathcal{FLE}$, in [48, 49] he defines a formal context called $\mathcal{FLE}$-*context*. The attributes of this formal context are $\mathcal{FLE}$-concept descriptions, and the objects are the elements of the domain over which these concept descriptions are interpreted. In this context, an object g is in relation with an attribute m if and only if g is in the interpretation of m. Thus, an implication holds in this formal context if and only if in the given model the concept description resulting from the conjunction of the attributes in the premise of the implication is subsumed by the concept description formed from the conclusion. This is how implications in $\mathcal{FLE}$-contexts give rise to subsumption relationships between $\mathcal{FLE}$ concept descriptions.

In order to obtain *complete* knowledge about the subsumption relationships in the given model between arbitrary $\mathcal{FLE}$ concepts, Rudolph gives a multi-step exploration algorithm. In the first step of the algorithm, he starts with an $\mathcal{FLE}$-context whose attributes are the atomic concepts occurring in a knowledge base. In exploration step $i + 1$, he defines the set of attributes as the union of the set of attributes from the first step and the set of concept descriptions formed by universally quantifying all attributes of the context at step i w.r.t. all atomic roles, and the set of concept descriptions formed by existentially quantifying all concept intents of the context at step i w.r.t all atomic roles. Rudolph points out that, at an exploration step, there can be some concept descriptions in the attribute set that are equivalent, i.e., attributes that can be reduced. To this aim, he introduces a method that he calls *empiric attribute reduction*. In principle, it is possible to carry out infinitely many exploration steps, which means that the algorithm will not terminate. In order to guarantee termination, Rudolph restricts the number of exploration steps. After carrying out i steps of exploration, it is then possible to decide subsumption (w.r.t. the given model) between any $\mathcal{FLE}$ concept descriptions up to role depth i just by using the implication bases

obtained as a result of the exploration steps. In addition, he also characterizes the cases where finitely many steps are sufficient to acquire complete information for deciding subsumption between $\mathcal{FLE}$ concept descriptions with arbitrary role depth. Rudolph argues that his method can be used to support the knowledge engineers in designing, building and refining DL ontologies. This method has been implemented in the tool Relexo.[3]

Exploring Finite Models in the DL $\mathcal{EL}_{gfp}$ In [4] Baader and Distel have extended classical FCA in order to provide support for analyzing relational structures by using efficient FCA algorithms. In this approach the atomic attributes are replaced by complex formulae in some logical language, and data is represented using relational structures rather than just formal contexts. This extension is later instantiated with atrributes defined in the DL $\mathcal{EL}$, and with relational structures defined over a signature of unary and binary predicates, i.e., models for $\mathcal{EL}$. In this setting an implication corresponds to a GCI in $\mathcal{EL}$. This approach at the first sight seems to be very close to the approach introduced in [48, 49]. One of the main differences between these approaches is that in [4] the authors use one context with infinitely many complex attributes, whereas in [49] Rudolph uses an infinite family of contexts, each having finitely many attributes that are obtained by restricting the role depth of concepts. In [4] the authors additionally show that for the DLs $\mathcal{EL}$ and $\mathcal{EL}_{gfp}$, which extends $\mathcal{EL}$ with cyclic concept definitions interpreted with *greatest fixpoint* semantics, the set of GCIs holding in a finite model always has a finite basis. That is, there is always a finite subset of the infinitely many GCIs from which the rest follows. Later in [5] the authors have shown how to compute this basis efficiently by using methods from FCA. In a follow-up paper [22], Distel has described how this method can be modified to allow ABox individuals as counterexamples to GCIs.

4 Contributions to combining DLs and FCA

Our contribution to the DL research by means of FCA methods falls mainly under two topics: 1) supporting bottom-up construction of DL knowledge bases, 2) completing DL knowledge bases. In Section 4.1 we briefly describe the use of FCA in the former, and in Section 4.2 we briefly describe the use of FCA in the latter contribution.

4.1 Supporting bottom-up construction of DL Ontologies

Traditionally, DL knowledge bases are built in a top-down manner, in the sense that first the relevant notions of the domain are formalized by concept descriptions, and then these concept descriptions are used to specify properties of the individuals occurring in the domain. However, this top-down approach is not

---

[3] http://relexo.ontoware.org

always adequate. On the one hand, it might not always be intuitive which notions of the domain are the relevant ones for a particular application. On the other hand, even if this is intuitive, it might not always be easy to come up with a clear formal description of these notions, especially for a domain expert who is not an expert in knowledge engineering. In order to overcome this, in [8] a new approach, called "bottom-up approach", was introduced for constructing DL knowledge bases. In this approach, instead of directly defining a new concept, the domain expert introduces several typical examples as objects, which are then automatically generalized into a concept description by the system. This description is then offered to the domain expert as a possible candidate for a definition of the concept. The task of computing such a concept description can be split into two subtasks:

- computing the most specific concepts of the given objects,
- and then computing the least common subsumer of these concepts.

The *most specific concept* (msc) of an object o is the most specific concept description C expressible in the given DL language that has o as an instance. The *least common subsumer (lcs)* of concept descriptions $C_1, \ldots, C_n$ is the most specific concept description C expressible in the given DL language that subsumes $C_1, \ldots, C_n$. The problem of computing the lcs and (to a more limited extent) the msc has already been investigated in the literature [8, 39, 2].

The methods for computing the least common subsumer are restricted to rather inexpressive descriptions logics not allowing for disjunction (and thus not allowing for full negation). In fact, for languages with disjunction, the lcs of a collection of concepts is just their disjunction, and nothing new can be learned from building it. In contrast, for languages without disjunction, the lcs extracts the "commonalities" of the given collection of concepts. Modern DL systems like FaCT++ [33, 57], RACER [29], Pellet [54], and Hermit [41] are based on very expressive DLs, and there exist large knowledge bases that use this expressive power and can be processed by these systems. In order to allow the user to re-use concepts defined in such existing knowledge bases and still support the user in defining new concepts with the bottom-up approach sketched above, in [15, 14, 16] we have proposed the following *extended bottom-up approach*: assume that there is a fixed *background terminology* defined in an expressive DL; e.g., a large ontology written by experts, which the user has bought from some ontology provider. The user then wants to extend this terminology in order to adapt it to the needs of a particular application domain. However, since the user is not a DL expert, he employs a less expressive DL and needs support through the bottom-up approach when building this user-specific extension of the background terminology. There are several reasons for the user to employ a restricted DL in this setting: first, such a restricted DL may be easier to comprehend and use for a non-expert; second, it may allow for a more intuitive graphical or frame-like user interface; third, to use the bottom-up approach, the lcs must exist and make sense, and it must be possible to compute it with reasonable effort.

To make this more precise, consider a background terminology (TBox) $\mathcal{T}$ defined in an expressive DL $\mathcal{L}_2$. When defining new concepts, the user employs

only a sublanguage $\mathcal{L}_1$ of $\mathcal{L}_2$, for which computing the lcs makes sense. However, in addition to primitive concepts and roles, the concept descriptions written in the DL $\mathcal{L}_1$ may also contain names of concepts defined in $\mathcal{T}$. Let us call such concept descriptions $\mathcal{L}_1(\mathcal{T})$-concept descriptions. Given $\mathcal{L}_1(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, we want to compute their lcs in $\mathcal{L}_1(\mathcal{T})$, i.e., the least $\mathcal{L}_1(\mathcal{T})$-concept description that subsumes $C_1, \ldots, C_n$ w.r.t. $\mathcal{T}$. In [14, 16] we have considered the case where $\mathcal{L}_1$ is the DL $\mathcal{ALE}$ and $\mathcal{L}_2$ is the DL $\mathcal{ALC}$, and shown the following result:

– If $\mathcal{T}$ is an acyclic $\mathcal{ALC}$-TBox, then the lcs w.r.t. $\mathcal{T}$ of $\mathcal{ALE}(\mathcal{T})$-concept descriptions always exists.

Unfortunately, the proof of this result does not yield a practical algorithm. Due to this, in [14, 16, 53] we have developed a more practical approach. Assume that $\mathcal{L}_1$ is a DL for which least common subsumers (without background TBox) always exist. Given $\mathcal{L}_1(\mathcal{T})$-concept descriptions $C_1, \ldots, C_n$, one can compute a common subsumer w.r.t. $\mathcal{T}$ by just ignoring $\mathcal{T}$, i.e., by treating the defined names in $C_1, \ldots, C_n$ as primitive and computing the lcs of $C_1, \ldots, C_n$ in $\mathcal{L}_1$. However, the common subsumer obtained this way will usually be too general. In [14, 16, 53], work we presented a method for computing "good" common subsumers w.r.t. background TBoxes, which may not be the *least* common subsumers, but which are better than the common subsumers computed by ignoring the TBox. In the present work we do not give the gcs algorithm in detail. We only demonstrate it on an example. The algorithm is described in detail in [16].

Example 1. As a simple example, consider the $\mathcal{ALC}$-TBox $\mathcal{T}$:

$$\mathsf{NoSon} \equiv \forall\mathsf{has\text{-}child}.\mathsf{Female},$$
$$\mathsf{NoDaughter} \equiv \forall\mathsf{has\text{-}child}.\neg\mathsf{Female},$$
$$\mathsf{SonRichDoctor} \equiv \forall\mathsf{has\text{-}child}.(\mathsf{Female} \sqcup (\mathsf{Doctor} \sqcap \mathsf{Rich})),$$
$$\mathsf{DaughterHappyDoctor} \equiv \forall\mathsf{has\text{-}child}.(\neg\mathsf{Female} \sqcup (\mathsf{Doctor} \sqcap \mathsf{Happy})),$$
$$\mathsf{ChildrenDoctor} \equiv \forall\mathsf{has\text{-}child}.\mathsf{Doctor},$$

and the $\mathcal{ALE}$-concept descriptions

$$C := \exists\mathsf{has\text{-}child}.(\mathsf{NoSon} \sqcap \mathsf{DaughterHappyDoctor}),$$
$$D := \exists\mathsf{has\text{-}child}.(\mathsf{NoDaughter} \sqcap \mathsf{SonRichDoctor}).$$

By ignoring the TBox, we obtain the $\mathcal{ALE}(\mathcal{T})$-concept description $\exists\mathsf{has\text{-}child}.\top$ as a common subsumer of C, D. However, if we take into account that both $\mathsf{NoSon} \sqcap \mathsf{DaughterHappyDoctor}$ and $\mathsf{NoDaughter} \sqcap \mathsf{SonRichDoctor}$ are subsumed by the concept $\mathsf{ChildrenDoctor}$, then we obtain the more specific common subsumer $\exists\mathsf{has\text{-}child}.\mathsf{ChildrenDoctor}$. The gcs of C, D is even more specific. In fact, the least conjunction of (negated) concept names subsuming both $\mathsf{NoSon} \sqcap \mathsf{DaughterHappyDoctor}$ and $\mathsf{NoDaughter} \sqcap \mathsf{SonRichDoctor}$ is

$$\mathsf{ChildrenDoctor} \sqcap \mathsf{DaughterHappyDoctor} \sqcap \mathsf{SonRichDoctor},$$

and thus the gcs of C, D is

$$\exists \mathsf{has\text{-}child}.(\mathsf{ChildrenDoctor} \sqcap \mathsf{DaughterHappyDoctor} \sqcap \mathsf{SonRichDoctor}).$$

The conjunct ChildrenDoctor is actually redundant since it is implied by the remainder of the conjunction. ◇

In order to implement the gcs algorithm, we must be able to compute the smallest conjunction of (negated) concept names that subsumes two such conjunctions C_1 and C_2 w.r.t. $\mathcal{T}$. In principle, one can compute this smallest conjunction by testing, for every (negated) concept name whether it subsumes both C_1 and C_2 w.r.t. $\mathcal{T}$, and then take the conjunction of those (negated) concept names for which the test was positive. However, this results in a large number of (possibly expensive) calls to the subsumption algorithm for $\mathcal{L}_2$ w.r.t. (general or (a)cyclic) TBoxes. Since, in our application scenario (bottom-up construction of DL knowledge bases w.r.t. a given background terminology), the TBox $\mathcal{T}$ is assumed to be fixed, it makes sense to precompute this information.

This is where FCA comes into play. By using the attribute exploration method [24] (possibly with background knowledge [25–27]), we compute the abovementioned smallest conjunction, which is required for computing a gcs. To this purpose we define a formal context whose concept lattice is isomorphic to the subsumption hierarchy we are interested in. In general, the subsumption relation induces a partial order, and not a lattice structure on concepts. However, in the case of conjunctions of (negated) concept names, all infima exist, and thus also all suprema, i.e., this hierarchy is a complete lattice. The experimental results in [16] have shown that the use of this hierarchy and its use in gcs computation are indeed quite efficient.

4.2 Completing DL Ontologies

The standardization of OWL [34] as the ontology language for the semantic web [17] led to the fact that several ontology editors like Protégé [38], and Swoop [37] now support OWL, 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 [51, 36, 35, 32, 11, 43]. 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 [6, 7] we have considered a different quality dimension: *completeness*. 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 to extend the ontology appropriately if this is not the case.

As already mentioned, 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 (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 describing it, we can ask whether the knowledge base 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 [59]. 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 have 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 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.

The attribute exploration method of FCA has proved to be a successful knowledge acquisition method in various application domains. One of the earliest applications of this approach is described in [58], 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 by 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 case, we cannot directly use this approach. The main reason is the open-world semantics of description logic knowledge bases. Consider an individual i from an ABox $\mathcal{A}$ and a concept C occurring in a TBox $\mathcal{T}$. If we cannot deduce from the TBox $\mathcal{T}$ and $\mathcal{A}$ that i is an instance of C, then we do not assume that i does not belong to C. Instead, we only accept this as a consequence if $\mathcal{T}$ and $\mathcal{A}$ imply that i is an instance of $\neg C$. Thus, our knowledge about the relationships between individuals and concepts is incomplete: if $\mathcal{T}$ and $\mathcal{A}$ imply neither $C(i)$ nor $\neg C(i)$, then we do not know the relationship between i and C. In contrast, classical FCA and attribute exploration assume that the knowledge about objects is complete: a cross in row g and column m of a formal context says that object g has attribute m, and the absence of a cross is interpreted as saying that g does not have m.

There has been some work on how to extend FCA and attribute exploration from complete knowledge to the case of partial knowledge [25, 18, 30, 31, 19, 49], and how to evaluate formulas in formal contexts that do not contain complete information [42]. However, these works are based on assumptions that are different from ours. In particular, they assume 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 knowledge base, 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 [7, 53] we have introduced an extension of FCA that can deal with partial knowledge. This extension is based on the notion of a *partial context* that consists of a set of *partial object descriptions* (*pod*). A pod is a tuple (A, S) where A represents the set of attributes that the pod is known to have, and S represents the set of attributes that the pod is known not to have. A and S are disjoint and their union need not be the whole attribute set, i.e., for some attributes it might be unknown whether the pod has this attribute or not. We say that a pod (A, S) *refutes* an implication $L \to R$ if $L \subseteq A$ and $R \cap S \neq \emptyset$. We also say that a partial context refutes an implication if there is a pod in this partial context that refutes this implication. Based on these, we define the notion of an *undecided implication*, which is an implication that does not follow from a given set of implications, and that is not refuted by a partial context. Then the attribute exploration method for partial contexts can be formulated as enumerating undecided implications as efficient as possible. In [7, 53] we have described a version of attribute exploration algorithm that works for this setting, and proved that this algorithm terminates and it is correct. Later we have shown that given a DL knowledge base $(\mathcal{T}, \mathcal{A})$, any individual in $\mathcal{A}$ gives rise to a pod, and thus $\mathcal{A}$ induces a partial context. This enables us to use our attribute exploration algorithm on partial contexts for

$\mathcal{A}_{countries}$	Asian	EU	European	G8	Mediterranean
Syria	+	-	-	-	+
Turkey	+	-	+	-	+
France	-	+	+	+	+
Germany	-	+	+	+	-
Switzerland	-	-	+	-	-
USA	-	-	-	+	-

Table 1. The partial context before completion

finding completing DL knowledge bases. As a result of running this algorithm on a DL knowledge base, the knowledge base is complete w.r.t. an intended interpreation, i.e., if an implication holds in this interpretation then it also follows from the TBox, and if not then the ABox contains a counterexample to this implication. For details of the attribute exploration on partial contexts and its application to DL ontologies we refer the reader to [7, 53], and demonstrate on a small example how it works.

Example 2. Let our TBox $\mathcal{T}_{countries}$ contain the following concept definitions:

$$\text{AsianCountry} \equiv \text{Country} \sqcap \exists \text{hasTerritoryIn}.\{Asia\}$$
$$\text{EUmember} \equiv \text{Country} \sqcap \exists \text{memberOf}.\{EU\}$$
$$\text{EuropeanCountry} \equiv \text{Country} \sqcap \exists \text{hasTerritoryIn}.\{Europe\}$$
$$\text{G8member} \equiv \text{Country} \sqcap \exists \text{memberOf}.\{G8\}$$
$$\text{IslandCountry} \equiv \text{Country} \sqcap \neg \exists \text{hasTerritoryIn}.\text{Continent}$$
$$\text{MediterrenaenCountry} \equiv \text{Country} \sqcap \exists \text{hasBorderTo}.\{MediterrenaenSea\}$$

Moreover, let our ABox $\mathcal{A}_{countries}$ contain the individuals *Syria, Turkey, France, Germany, Switzerland, USA* and assume we are interested in the subsumption relationships between the concept names AsianCountry, EUmember, European-Country, G8member and MediterreneanCountry. Table 1 shows the partial context induced by $\mathcal{A}_{countries}$, and Table 2 shows the questions asked by the completion algorithm and the answers given to these questions. In order to save space, the names of the concepts are shortened in both tables. The questions with positive answers result in extension of the TBox with the following GCIs:

$$\text{G8member} \sqcap \text{MediterraneanCountry} \sqsubseteq \text{EUmember} \sqcap \text{EuropeanCountry}$$
$$\text{EUmember} \sqcap \text{G8member} \sqsubseteq \text{EuropeanCountry}$$
$$\text{AsianCountry} \sqcap \text{EUmember} \sqsubseteq \text{MediterraneanCountry}$$
$$\text{AsianCountry} \sqcap \text{EUmember} \sqcap$$
$$\text{EuropeanCountry} \sqcap \text{MediterraneanCountry} \sqsubseteq \text{G8member}$$

Moreover, the questions with negative answers result in extension of the ABox with the individuals *Russia, Cyprus, Spain* and *Japan*. The partial context induced by the resulting ABox $\mathcal{A}'_{countries}$ is shown in Table 3. The resulting

Question	Answer	Counterex.
{G8, Mediterranean} → {EU, European}?	yes	-
{European, G8} → {EU}?	no	Russia
{EU} → {European, G8}?	no	Cyprus
{EU, G8} → {European}?	yes	-
{EU, European} → {G8}?	no	Spain
{Asian, G8} → {European}?	no	Japan
{Asian, EU} → {Mediterranean}?	yes	-
{Asian, EU, European, Mediterranean} → {G8}?	yes	-

Table 2. Execution of the ontology completion algorithm $(\mathcal{T}_{countries}, \mathcal{A}_{countries})$

$\mathcal{A}'_{countries}$	Asian	EU	European	G8	Mediterranean
Syria	+	-	-	-	+
Turkey	+	-	+	-	+
France	-	+	+	+	+
Germany	-	+	+	+	-
Switzerland	-	-	+	-	-
USA	-	-	-	+	-
Russia	+	-	+	+	-
Cyprus	+	+	-	-	+
Spain	-	+	+	-	+
Japan	+	-	-	+	-

Table 3. The partial context after completion

knowledge base $(\mathcal{T}'_{countries}, \mathcal{A}'_{countries})$ is complete w.r.t. the initially selected concept names.

$\diamond$

Based on on the described approach, we implemented a first experimental version of a DL knowledge base completion tool as an extension for the Swoop ontology editor using Pellet as the underlying reasoner. A first evaluation of this tool on the OWL ontology for human protein phosphatases 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 completion process from scratch. Another usability issue on the wish list of our experts was to allow the postponement of answering certain questions, while continuing the completion process with other questions.

In a follow-up paper [13] we have addressed these usability issues. We have improved the method in such a way that at any time during completion the expert can pause the process, see all of her previous answers or changes to the knowledge base, 'undo' some of those changes, and continue completion. Here

we of course paid attention that the expert does not have to answer the same questions she has answered before pausing the process. We have achieved this by saving previous answers, and using them as background knowledge when the expert continues completion. The other wish of our experts, namely postponing questions was solved pausing completion, changing the order of attributes, and restarting the completion with previous answers as background knowledge. In theory, this method might not postpone a question, thus the expert might be asked the last question again. However, in practice the method turned out to be useful in many cases when the expert was not able to answer a particular question and wanted to get another one. We have implemented our ontology completion method together with these usability issues as a plugin for the Protégé ontology editor under the name OntoComp.[4]

5   Conclusion

We have summarized the work done in combining DLs and FCA. The research done in this field mainly falls under two categories: 1) efforts to enrich the language of FCA by borrowing constructors from DL languages, and 2) efforts to employ FCA methods in the solution of problems encountered in knowledge representation with DLs. For each of these categories we have given pointers and shortly described the relevant work in the literature. We have also described our own contributions, which are mainly under the second category.

Recent developments in information technologies like social networks, Web 2.0 applications and semantic web applications are bringing up new challenges for representing vast amounts of knowledge and analyzing huge amounts of data rapidly generated by these applications. The two research areas we have discussed here, namely DLs and FCA, are lying at the core of representing knowledge, and analyzing data, respectively. We are confident that these new challenges will enable new fruitful cooperations between these two research fields.

References

1. F. Baader. Computing a minimal representation of the subsumption lattice of all conjunctions of concepts defined in a terminology. In G. Ellis, R. A. Levinson, A. Fall, and V. Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proceedings of the 1st International KRUSE Symposium*, pages 168–178, 1995.
2. F. Baader. Least common subsumers and most specific concepts in a description logic with existential restrictions and terminological cycles. In G. Gottlob and T. Walsh, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 319–324. Morgan Kaufmann, 2003.
3. 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.

---

[4] `http://ontocomp.googlecode.com`

4. F. Baader and F. Distel. A finite basis for the set of EL-implications holding in a finite model. In R. Medina and S. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis, (ICFCA 2008)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 46–61. Springer-Verlag, 2008.

5. F. Baader and F. Distel. Exploring finite models in the description logic ELgfp. In S. Ferré and S. Rudolph, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Intelligence*, pages 146–161. Springer-Verlag, 2009.

6. F. Baader, B. Ganter, U. Sattler, and B. Sertkaya. Completing description logic knowledge bases using formal concept analysis. In *Proceedings of the Third International Workshop OWL: Experiences and Directions (OWLED 2007)*. CEUR-WS, 2007.

7. F. Baader, B. Ganter, B. Sertkaya, and U. Sattler. Completing description logic knowledge bases using formal concept analysis. In M. M. Veloso, editor, *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07)*, pages 230–235. AAAI Press, 2007.

8. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.

9. F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer-Verlag, 2006.

10. F. Baader and R. Molitor. Building and structuring description logic knowledge bases using least common subsumers and concept analysis. In B. Ganter and G. W. Mineau, editors, *Proceedings of the 8th International Conference on Conceptual Structures (ICCS 2000)*, volume 1867 of *Lecture Notes in Computer Science*, pages 292–305. Springer-Verlag, 2000.

11. F. Baader and R. Peñaloza. Axiom pinpointing in general tableaux. *Journal of Logic and Computation*, 2010. To appear.

12. F. Baader and B. Sertkaya. Applying formal concept analysis to description logics. In P. Eklund, editor, *Proceedings of the 2nd International Conference on Formal Concept Analysis (ICFCA 2004)*, volume 2961 of *Lecture Notes in Computer Science*, pages 261–286, Sydney, Australia, 2004. Springer-Verlag.

13. F. Baader and B. Sertkaya. Usability issues in description logic knowledge base completion. In S. Ferré and S. Rudolph, editors, *Proceedings of the 7th International Conference on Formal Concept Analysis, (ICFCA 2009)*, volume 5548 of *Lecture Notes in Artificial Intelligence*, pages 1–21. Springer-Verlag, 2009.

14. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. In J. J. Alferes and J. A. Leite, editors, *Proceedings of the 9th European Conference on Logics in Artificial Intelligence (JELIA 2004)*, volume 3229 of *Lecture Notes in Computer Science*, pages 400–412, Lisbon, Portugal, 2004. Springer-Verlag.

15. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. In V. Haarslev and R. Möller, editors, *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, volume 104 of *CEUR Workshop Proceedings*, Whistler, Canada, 2004. CEUR-WS.org.

16. F. Baader, B. Sertkaya, and A.-Y. Turhan. Computing the least common subsumer w.r.t. a background terminology. *Journal of Applied Logic*, 5(3), 2007.

17. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, 284(5):34–43, 2001.
18. P. Burmeister and R. Holzer. On the treatment of incomplete knowledge in formal concept analysis. In B. Ganter and G. W. Mineau, editors, *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.
19. 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.
20. A. Coulet, M. Smaïl-Tabbone, A. Napoli, and M.-D. Devignes. Ontology refinement through role assertion analysis: Example in pharmacogenomics. In F. Baader, C. Lutz, and B. Motik, editors, *Proceedings of the 21st International Workshop on Description Logics, (DL2008)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
21. A. Coulet, M. Smaïl-Tabbone, A. Napoli, and M.-D. Devignes. Role assertion analysis: a proposed method for ontology refinement through assertion learning. In *Proceedings of the Fourth Starting AI Researchers' Symposium, (STAIRS 2008)*, volume 179 of *Frontiers in Artificial Intelligence and Applications*, pages 47–58. IOS Press, 2008.
22. F. Distel. An approach to exploring description logic knowledge bases. In L. Kwuida and B. Sertkaya, editors, *Proceedings of the 8th International Conference on Formal Concept Analysis, (ICFCA 2010)*, volume 5986 of *Lecture Notes in Artificial Intelligence*, pages 209–224. Springer-Verlag, 2010.
23. S. Ferré, O. Ridoux, and B. Sigonneau. Arbitrary relations in formal concept analysis and logical information systems. In F. Dau, M.-L. Mugnier, and G. Stumme, editors, *Proceedings of the 13th International Conference on Conceptual Structures, (ICCS 2005)*, volume 3596 of *Lecture Notes in Computer Science*, pages 166–180. Springer-Verlag, 2005.
24. B. Ganter. Two basic algorithms in concept analysis. Technical Report Preprint-Nr. 831, Technische Hochschule Darmstadt, Darmstadt, Germany, 1984.
25. B. Ganter. Attribute exploration with background knowledge. *Theoretical Computer Science*, 217(2):215–233, 1999.
26. B. Ganter and R. Krauße. Pseudo models and propositional Horn inference. Technical Report MATH-AL-15-1999, Institut für Algebra, Technische Universität Dresden, Dresden, Germany, 1999.
27. B. Ganter and R. Krauße. Pseudo-models and propositional Horn inference. *Discrete Applied Mathematics*, 147(1):43–55, 2005.
28. B. Ganter and R. Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer-Verlag, Berlin, Germany, 1999.
29. V. Haarslev and R. Möller. RACER system description. In *Proceedings International Joint Conference on Automated Reasoning (IJCAR 2001)*, pages 701–706, 2001.
30. R. Holzer. Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part I. *Fundamenta Informaticae*, 63(1):17–39, 2004.
31. R. Holzer. Knowledge acquisition under incomplete knowledge using methods from formal concept analysis: Part II. *Fundamenta Informaticae*, 63(1):41–63, 2004.
32. M. Horridge, B. Parsia, and U. Sattler. Laconic and precise justifications in owl. In A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. W. Finin, and K. Thirunarayan, editors, *Proceedings of the 7th International Semantic Web Conference, (ISWC 2008)*, volume 5318 of *Lecture Notes in Computer Science*, pages 323–338. Springer-Verlag, 2008.

33. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proceedings of the 6th International Conference on the Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

34. 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.

35. A. Kalyanpur, B. Parsia, M. Horridge, and E. Sirin. Finding all justifications of OWL DL entailments. In *Proceedings of the 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, (ISWC 2007 + ASWC 2007)*, volume 4825 of *Lecture Notes in Computer Science*, pages 267–280. Springer-Verlag, 2007.

36. A. Kalyanpur, B. Parsia, E. Sirin, and B. C. Grau. Repairing unsatisfiable concepts in OWL ontologies. In Y. Sure and J. Domingue, editors, *The Semantic Web: Research and Applications. Proceedings of the 3rd European Semantic Web Conference (ESWC 2006)*, volume 4011 of *Lecture Notes in Computer Science*, pages 170–184. Springer-Verlag, 2006.

37. A. Kalyanpur, B. Parsia, E. Sirin, B. C. Grau, and J. A. Hendler. Swoop: A web ontology editing browser. *Journal of Web Semantics*, 4(2):144–153, 2006.

38. 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 S. A. McIlraith, D. Plexousakis, and F. van Harmelen, editors, *Proceedings of the 3rd International Semantic Web Conference, (ISWC 2004)*, volume 3298 of *Lecture Notes in Computer Science*, pages 229–243. Springer-Verlag, 2004.

39. R. Küsters and R. Molitor. Computing least common subsumers in $\mathcal{ALEN}$. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, (IJCAI 2001)*, pages 219–224. Morgan Kaufmann, 2001.

40. B. Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. Ph.D. dissertation, Universität Karlsruhe (TH), Germany, 2006.

41. B. Motik, R. Shearer, and I. Horrocks. Hypertableau Reasoning for Description Logics. *Journal of Artificial Intelligence Research*, 36:165–228, 2009.

42. 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.

43. R. Peñaloza and B. Sertkaya. On the complexity of axiom pinpointing in the $\mathcal{EL}$ family of Description Logics. In F. Lin and U. Sattler, editors, *Proceedings of the Twelfth International Conference on Principles and Knowledge Representation and Reasoning (KR-10)*. Morgan Kaufmann, 2010.

44. S. Prediger. Terminologische Merkmalslogik in der Formalen Begriffsanalyse. In G. Stumme and R. Wille, editors, *Begriffliche Wissensverarbeitung – Methoden und Anwendungen*, pages 99–124, Heidelberg, Germany, 2000. Springer-Verlag.

45. S. Prediger and G. Stumme. Theory-driven logical scaling: Conceptual information systems meet description logics. In E. Franconi and M. Kifer, editors, *Proceedings of the 6th International Workshop on Knowledge Representation meets Databases (KRDB'99)*, 1999.

46. M. H. Rouane, M. Huchard, A. Napoli, and P. Valtchev. A proposal for combining formal concept analysis and description logics for mining relational data. In S. O. Kuznetsov and S. Schmidt, editors, *Proceedings of the 5th International Conference on Formal Concept Analysis, (ICFCA 2007)*, volume 4390 of *Lecture Notes in Computer Science*, pages 51–65. Springer-Verlag, 2007.

47. S. Rudolph. An FCA method for the extensional exploration of relational data. In B. Ganter and A. de Moor, editors, *Contributions to International Conference on Conceptual Structures 2003 (ICCS 2003)*, pages 197–210. Shaker Verlag, 2003.
48. S. Rudolph. Exploring relational structures via $\mathcal{FLE}$. In K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, editors, *Proceedings of the 12th International Conference on Conceptual Structures (ICCS 2004)*, volume 3127 of *Lecture Notes in Computer Science*, pages 196–212. Springer-Verlag, 2004.
49. S. Rudolph. *Relational exploration: Combining Description Logics and Formal Concept Analysis for knowledge specification.* Ph.D. dissertation, Fakultät Mathematik und Naturwissenschaften, TU Dresden, Germany, 2006.
50. S. Rudolph. Acquiring generalized domain-range restrictions. In R. Medina and S. Obiedkov, editors, *Proceedings of the 6th International Conference on Formal Concept Analysis, (ICFCA 2008)*, volume 4933 of *Lecture Notes in Artificial Intelligence*, pages 32–45, 2008.
51. S. Schlobach and R. Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In G. Gottlob and T. Walsh, editors, *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, pages 355–362. Morgan Kaufmann, 2003.
52. B. Sertkaya. Computing the hierarchy of conjunctions of concept names and their negations in a description logic knowledge base using formal concept analysis. In *Supplementary Proceedings of the 4th International Conference on Formal Concept Analysis, (ICFCA 2006)*, Dresden, Germany, 2006.
53. B. Sertkaya. *Formal Concept Analysis Methods for Description Logics.* Ph.D. dissertation, Institute of Theoretical Computer Science, TU Dresden, Germany, 2007.
54. E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proceedings of the 2004 International Workshop on Description Logics (DL2004)*, volume 104 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2004.
55. G. Stumme. The concept classification of a terminology extended by conjunction and disjunction. In N. Y. Foo and R. Goebel, editors, *Proceedings of the 4th Pacific Rim International Conference on Artificial Intelligence (PRICAI'96)*, volume 1114 of *Lecture Notes in Computer Science*, pages 121–131. Springer-Verlag, 1996.
56. G. Stumme. Distributive concept exploration - a knowledge acquisition tool in formal concept analysis. In O. Herzog and A. Günter, editors, *Proceedings of the 22nd Annual German Conference on Artificial Intelligence (KI'98)*, volume 1504 of *Lecture Notes in Computer Science*, pages 117–128. Springer-Verlag, 1998.
57. D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In U. Furbach and N. Shankar, editors, *Proceedings of the International Joint Conference on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer-Verlag, 2006.
58. 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.
59. 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 *Proceedings of the 4th International Semantic Web Conference, (ISWC 2005)*, volume 3729 of *Lecture Notes in Computer Science*, pages 786–800. Springer-Verlag, 2005.
60. M. Zickwolff. *Rule Exploration: First Order Logic in Formal Concept Analysis.* Ph.D. dissertation, TH Darmstadt, Germany, 1991.