Small is Again Beautiful in Description Logics

Franz Baader, Carsten Lutz, Anni-Yasmin Turhan

The Description Logic (DL) research of the last 20 years was mainly concerned with increasing the expressive power of the employed description language without losing the ability of implementing highly-optimized reasoning systems that behave well in practice, inspite of the ever increasing worst-case complexity of the underlying inference problems. OWL DL, the standard ontology language for the Semantic Web, is based on such an expressive DL for which reasoning is highly intractable. Its sublanguage OWL Lite was intended to provide a tractable version of OWL, but turned out to be only of a slightly lower worst-case complexity than OWL DL. This and other reasons have led to the development of two new families of light-weight DLs, $\mathcal{EL}$ and DL-Lite, which recently have been proposed as profiles of OWL 2, the new version of the OWL standard. In this paper, we give an introduction to these new logics, explaining the rationales behind their design.

1 Introduction

Description Logics [8] are a well-investigated 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, and databases, but their most notable success so far is the adoption of the DL-based language OWL[1] as a standard ontology language for the Semantic Web [33, 11].

In DLs, concepts are formally described by *concept descriptions*, i.e., expressions that are built from concept names (unary predicates) and role names (binary predicates) using concept constructors. The expressivity of a particular DL is determined by which concept constructors are available in it. From a semantic point of view, concept names and concept descriptions represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the concept name *Woman*, and the role name *child*, the concept of *women having a daughter* can be represented by the concept description

$$\mathit{Woman} \sqcap \exists \mathit{child}.\mathit{Woman},$$

and the concept of *women having only daughters* by

$$\mathit{Woman} \sqcap \forall \mathit{child}.\mathit{Woman}.$$

In its simplest form, a DL *terminology* (usually called *TBox*) can be used to introduce abbreviations for complex concept descriptions. For example, the *concept definitions*

$$\begin{array}{rcl} \mathit{Woman} & \equiv & \mathit{Human} \sqcap \mathit{Female} \\ \mathit{Mother} & \equiv & \mathit{Woman} \sqcap \exists \mathit{child}.\top \end{array}$$

define the concept of a woman as a human that is female, and the concept of a mother as a woman that has a child, where $\top$ stands for the top concept (which is interpreted as the universe of all individuals in the application domain). So-called *general concept inclusions (GCIs)* can be used to state additional constraints on the interpretation of concepts and roles. In our example, it makes sense to state domain and range restrictions for the role *child*. The GCIs

$$\begin{array}{rcl} \exists \mathit{child}.\mathit{Human} & \sqsubseteq & \mathit{Human} \\ \mathit{Human} & \sqsubseteq & \forall \mathit{child}.\mathit{Human} \end{array}$$

respectively say that only human beings can have human children, and that the child of a human being must be human.

In the *assertional part (ABox)* of a DL knowledge base, facts about a specific application situation can be stated, by introducing named individuals and relating them to concepts and roles. For example, the assertions

$$\mathit{Woman}(\mathit{LINDA}), \quad \mathit{child}(\mathit{LINDA}, \mathit{JAMES})$$

state that Linda is a woman, who has the child James.

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the *subsumption* service allows one to determine subconcept-superconcept relationships. For example, w.r.t. the concept definitions from above, the concept *Female* subsumes the concept *Mother* since all instances of the second concept are necessarily instances of the first concept, i.e., whenever the above concept definitions are satisfied, then *Mother* is interpreted as a subset of *Female*. With the help of the subsumption service, one can compute the hierarchy of all concepts defined in a TBox. This compound inference service is usually called *classification*. The *instance* service can be used to check whether an individual occurring in an ABox is necessarily an instance of a given concept. For example, w.r.t. the above assertions, concept definitions, and GCIs, the individual *JAMES* is an instance of the concept *Human*. With the help of the instance service, one can also compute answers to *instance queries*, i.e., all individuals occurring in the ABox that are instances of the query concept C. In order to state more general search criteria, one can use so-called *conjunctive queries*, i.e., conjunctions of assertions that may also contain variables, of which some can be existentially quantified. For example, the conjunctive query

$$\exists y, z.\mathit{Woman}(x) \wedge \mathit{child}(x, y) \wedge \mathit{child}(z, y) \wedge \mathit{Beatle}(z)$$

asks for all women that have a child with a parent that is a Beatle. With respect to the knowledge base we have introduced so far, this conjunctive query has no individual as an answer.

[1]`http://www.w3.org/TR/owl-features/`

In order to ensure a reasonable and predictable behavior of a DL system, the underlying inference problems (like the subsumption and the instance problem) should at least be decidable for the DL employed by the system, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues in DL research.

The general opinion on the (worst-case) complexity that is acceptable for a DL has changed dramatically over time. Historically, in the early times of DL research people concentrated on identifying formalisms for which reasoning is tractable, i.e., can be performed in polynomial time [47]. The precursor of all DL systems, KL-ONE [16], as well as its early successor systems, like KANDOR [47], K-REP [43], BACK [48], and LOOM [42], indeed employed polynomial-time subsumption algorithms. Later on, however, it turned out that subsumption in rather inexpressive DLs may be intractable [38], that subsumption in KL-ONE is even undecidable [49], and that even for systems like KANDOR and BACK, for which the expressiveness of the underlying DL had been carefully restricted with the goal of retaining tractability, the subsumption problem is in fact intractable [44]. The reason for the discrepancy between the complexity of the subsumption algorithms employed in the above mentioned early DL systems and the worst-case complexity of the subsumption problems these algorithms were supposed to solve was due to the fact that these systems employed sound, but incomplete subsumption algorithms, i.e., algorithms whose positive answers to subsumption queries are correct, but whose negative answers may be incorrect. The use of incomplete algorithms has since then largely been abandoned in the DL community, mainly because of the problem that the behavior of the systems is no longer determined by the semantics of the description language: an incomplete algorithm may claim that a subsumption relationship does not hold, although it should hold according to the semantics. All the intractability results mentioned above already hold for subsumption between concept descriptions without a TBox. An even worse blow to the quest for a practically useful DL with a sound, complete, and polynomial-time subsumption algorithm was Nebel's result [45] that subsumption w.r.t. an acyclic TBox (i.e., an unambiguous set of concept definitions without cyclic dependencies) in a DL with conjunction ($\sqcap$) and value restriction ($\forall r.C$) is already intractable.[2]

At about the time when these (negative) complexity results were obtained, a new approach for solving inference problems in DLs, such as the subsumption and the instance problem, was introduced. This so-called *tableau-based approach* was first introduced in the context of DLs by Schmidt-Schauß and Smolka [50], though it had already been used for modal logics long before that [22]. It has turned out that this approach can be used to handle a great variety of different DLs [27, 26, 10, 7, 35, 15, 34, 30], and it yields sound and complete inference algorithms also for very expressive DLs. Although the worst-case complexity of these algorithms is quite high, the tableau-based approach nevertheless often yields practical procedures: optimized implementations of such procedures have turned out to behave quite well in applications [9, 28, 31, 23, 29, 25], even for expressive DLs with a high worst-case complexity (ExpTime and beyond). The advent of efficient tableau-based algorithms was the main reason why the DL community basically abandoned the search for DLs with tractable inference problems, and concentrated on the design of practical tableau-based algorithms for expressive DLs. The most prominent modern DL systems, FaCT++ [53], Racer [24], and Pellet [51] support very expressive DLs and employ highly-optimized tableau-based algorithms.

In addition to the fact that DLs are equipped with a well-defined formal semantics, the availability of mature systems that support sound and complete reasoning in very expressive description formalisms was an important argument in favor of using DLs as the foundation of OWL, the standard ontology language for the Semantic Web. In fact, OWL DL is based on the expressive DL $\mathcal{SHOIN}(\mathcal{D})$, for which reasoning is NExpTime-complete, and its sublanguage OWL Lite is based on $\mathcal{SHIF}(\mathcal{D})$, for which reasoning is still ExpTime-complete [32]. The OWL 2 standard is based on the even more expressive DL $\mathcal{SROIQ}(\mathcal{D})$, which is even 2NExpTime-complete [36].

Due to the ever increasing expressive power and worst-case complexity of expressive DLs, there is also an increasing number of ontologies emerging from practical applications that cannot be handled by tableau-based reasoning systems without manual tuning by the system developers, despite highly optimized implementations. Perhaps the most prominent example is the well-known medical ontology SNOMED CT,[3] which comprises 380,000 concepts and is used to generate a standardized health care terminology used as a standard for medical data exchange in a variety of countries such as the US, Canada, and Australia. In tests performed in 2005 with FaCT++ and Racer, neither of the two systems could classify SNOMED CT [13],[4] and Pellet still could not classify SNOMED CT in tests performed in 2008 [52]. From the DL point of view, SNOMED CT is an acyclic TBox that contains only the concept constructors conjunction ($\sqcap$), existential restriction ($\exists r.C$), and the top concept ($\top$). The DL with exactly these three concept constructors is called $\mathcal{EL}$ [12]. In contrast to its counterpart with value restrictions, $\mathcal{FL}_0$, the light-weight DL $\mathcal{EL}$ has much better algorithmic properties. Whereas subsumption without a TBox is polynomial in both $\mathcal{EL}$ [12] and $\mathcal{FL}_0$ [38], subsumption in $\mathcal{FL}_0$ w.r.t. an acyclic TBox is coNP-complete [45] and w.r.t. GCIs it is even ExpTime-complete [5]. In contrast, subsumption in $\mathcal{EL}$ stays tractable even w.r.t. GCIs [17], and this result is stable under the addition of several interesting means of expressivity [5, 6]. The DL $\mathcal{EL}$ and the mentioned tractability results will be introduced in more detail in the next section.

Another issue with expressive DLs and tableau-based algorithms is that they do not scale too well to knowledge bases with a very large ABox. In particular, query answering in expressive DLs such as the already mentioned $\mathcal{SHIF}$ and $\mathcal{SHOIN}$ is 2ExpTime-complete regarding combined complexity [39], i.e., the complexity w.r.t. the size of the TBox and the ABox. Thus

[2] All the systems mentioned above supported these two concept constructors, which were at that time viewed as being indispensable for a DL. The DL with exactly these two concept constructors is called $\mathcal{FL}_0$ [4].

[3] `http://www.ihtsdo.org/snomed-ct/`

[4] Note, however, that more recent versions of FaCT++ and Racer perform quite well on SNOMED CT [52], due to optimizations specifically tailored towards the classification of SNOMED CT.

Name	Syntax	Semantics
concept name	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
role name	r	$r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
top concept	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
existential restriction	$\exists r.C$	$(\exists r.C)^{\mathcal{I}} = \{x \mid \exists y : (x,y) \in r^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
general concept inclusion (GCI)	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept definition	$A \equiv C$	$A^{\mathcal{I}} = C^{\mathcal{I}}$

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

query answering in these logics is even harder than subsumption while at the same time being much more time critical. Moreover, query answering in these DLs is coNP-complete [46] regarding data complexity (i.e., in the size of the ABox), which is viewed as 'unfeasible' in the database community. These results are dramatic since many DL applications, such as those that use ABoxes as kind of web repositories, involve ABoxes with hundred of thousands of individuals. It is a commonly held opinion that, in order to achieve truly scalable query answering in the short term, it is essential to make use of conventional relational database systems for query answering in DLs. Given this proviso, the question is what expressivity can a DL offer such that queries can be answered using relational database technology while at the same time meaningful concepts can be specified in the TBox. As an answer to this, the DL-Lite family has been introduced in [18, 19], designed to allow the implementation of conjunctive query answering 'on top of' a relational database system. In Section 3, we introduce DL-Lite$_{core}$ and two of its extensions DL-Lite$_{\mathcal{F}}$ and DL-Lite$_{\mathcal{R}}$. We also sketch the standard approach to query answering in these languages. Interestingly, also in $\mathcal{EL}$ it is possible to implement query answering using a database system, though with a different approach than in DL-Lite (see the end of Section 3).

2 The DL $\mathcal{EL}$ and its extension $\mathcal{EL}^{++}$

Starting with a set N_{con} of concept names and a set N_{role} of role names, *$\mathcal{EL}$-concept descriptions* are built using the concept constructors top concept ($\top$), conjunction ($\sqcap$), and existential restriction ($\exists r.C$). The semantics of $\mathcal{EL}$-concept descriptions is defined in the usual way, using the notion of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, which consists of a non-empty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that assigns binary relations on $\Delta^{\mathcal{I}}$ to role names and subsets of $\Delta^{\mathcal{I}}$ to concept descriptions, as shown in the semantics column of Table 1.

A *general concept inclusion (GCI)* is of the form $C \sqsubseteq D$ where C, D are $\mathcal{EL}$-concept descriptions, and a *concept definition* is of the form $A \equiv C$ where A is a concept name and C is an $\mathcal{EL}$-concept description. The interpretation $\mathcal{I}$ is a *model* of the GCI $C \sqsubseteq D$ or the concept definition $A \equiv C$ if it satisfies the condition stated in the semantics column of Table 1. Obviously, this semantics implies that the concept definition $A \equiv C$ is equivalent to the two GCIs $A \sqsubseteq C, C \sqsubseteq A$ in the sense that they have the same models. For this reason, in the following we will consider only GCIs. A finite set of GCIs is called a *TBox*.

```
(R1)  If    $A_1 \sqcap A_2 \sqsubseteq B \in \mathcal{T}$ and $A_1, A_2 \in S(A)$
      then  add $B$ to $S(A)$
(R2)  If    $A_1 \sqsubseteq \exists r.B \in \mathcal{T}$ and $A_1 \in S(A)$
      then  add $r$ to $R(A,B)$
(R3)  If    $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$ and $B_1 \in S(B), r \in S(A,B)$
      then  add $A_1$ to $S(A)$
```

Figure 1: The completion rules for subsumption in $\mathcal{EL}$.

Given a TBox $\mathcal{T}$ and two $\mathcal{EL}$-concept descriptions C, D, we say that C is *subsumed* by D w.r.t. $\mathcal{T}$ (written $C \sqsubseteq_{\mathcal{T}} D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for all models $\mathcal{I}$ of $\mathcal{T}$.[5]

When designing a subsumption algorithm for $\mathcal{EL}$ it is actually enough to consider the case where C, D are concept names occurring in the TBox. In fact, it is easy to see that $C \sqsubseteq_{\mathcal{T}} D$ iff $A \sqsubseteq_{\mathcal{T} \cup \{A \sqsubseteq C, D \sqsubseteq B\}} B$ where A, B are new concept names, i.e., concept names not occurring in C, D, and $\mathcal{T}$.

The polynomial-time subsumption algorithm for $\mathcal{EL}$ [17, 5] that will be sketched below actually classifies the given TBox $\mathcal{T}$, i.e., it simultaneously computes all subsumption relationships between the concept names occurring in $\mathcal{T}$. This algorithm proceeds in four steps:

1. Normalize the TBox.
2. Translate the normalized TBox into a graph.
3. Complete the graph using completion rules.
4. Read off the subsumption relationships from the normalized graph.

An $\mathcal{EL}$-TBox is *normalized* iff it only contains GCIs of the following form: $A_1 \sqcap A_2 \sqsubseteq B, A \sqsubseteq \exists r.B, \exists r.A \sqsubseteq B$, where A, A_1, A_2, B are concept names or the top concept $\top$. Any $\mathcal{EL}$-TBox can be transformed in polynomial time into a normalized one by applying equivalence-preserving normalization rules [17].

In the next step, we build the *classification graph* $G_{\mathcal{T}} = (V, V \times V, S, R)$ where

- V is the set of concept names (including $\top$) occurring in the normalized TBox $\mathcal{T}$;
- S labels nodes with sets of concept names (again including $\top$);
- R labels edges with sets of role names.

The label sets are supposed to satisfy the following *invariants*:

- $B \in S(A)$ implies $A \sqsubseteq_{\mathcal{T}} B$, i.e., $S(A)$ contains only subsumers of A w.r.t. $\mathcal{T}$.
- $r \in R(A,B)$ implies $A \sqsubseteq_{\mathcal{T}} \exists r.B$, i.e., $R(A,B)$ contains only roles r such that $\exists r.B$ subsumes A w.r.t. $\mathcal{T}$.

Initially, we set $S(A) := \{A, \top\}$ for all nodes $A \in V$, and $R(A,B) := \emptyset$ for all edges $(A,B) \in V \times V$. Obviously, the above invariants are satisfied by these initial label sets.

The labels of nodes and edges are then extended by applying the rules of Figure 1. Note that a rule is only applied if it really extends a label set. It is easy to see that these rules preserve the above invariants. For example, consider the (most complicated) rule (R3). Obviously, $\exists r.B_1 \sqsubseteq A_1 \in \mathcal{T}$ implies $\exists r.B_1 \sqsubseteq_{\mathcal{T}} A_1$, and the assumption that the invariants are satisfied before applying the rule yields $B \sqsubseteq_{\mathcal{T}} B_1$ and $A \sqsubseteq_{\mathcal{T}} \exists r.B$. The subsumption

[5]In this section, we do not introduce ABoxes and the instance problem. It should be noted, however, that the tractability results sketched in this section extend to the instance problem.

relationship $B \sqsubseteq_\mathcal{T} B_1$ obviously implies $\exists r.B \sqsubseteq_\mathcal{T} \exists r.B_1$. By applying transitivity of the subsumption relation $\sqsubseteq_\mathcal{T}$, we thus obtain $A \sqsubseteq_\mathcal{T} A_1$.

The fact that subsumption in $\mathcal{EL}$ w.r.t. TBoxes can be decided in polynomial time is an immediate consequence of the following two facts (see [17, 5] for proofs):

1. Rule application terminates after a polynomial number of steps.
2. If no more rules are applicable, then $A \sqsubseteq_\mathcal{T} B$ iff $B \in S(A)$.

Theorem 1 *Subsumption in $\mathcal{EL}$ w.r.t. TBoxes can be decided in polynomial time.*

This result is not only of theoretical interest. Experiments have shown that an optimized implementation [13] of the subsumption algorithm sketched above in the CEL system[6] [14] behaves very well on large life science ontologies [13, 52].

The tractability result for $\mathcal{EL}$ can be extended to $\mathcal{EL}^{++}$, which extends $\mathcal{EL}$ by the following means of expressiveness:

- The *bottom concept* $\bot$ is always interpreted as the empty set. It can, for example, be used to express disjointness of concepts, as in the GCI *Woman* $\sqcap$ *Man* $\sqsubseteq \bot$.
- *Nominals* are basically names for individuals, but used as concept constructors with set brackets around the individual name. A nominal $\{n\}$ is always interpreted as a singleton set. For example, we can use the nominal $\{\mathit{OBAMA}\}$ to express the concept of all individuals that like Obama: $\exists \mathit{likes}.\{\mathit{OBAMA}\}$. Nominals can also be used to express ABox assertions through GCIs. For example, the role assertion $r(a, b)$ can be expressed as $\{a\} \sqsubseteq \exists r.\{b\}$.
- *Concrete domains* can be used to refer to data types like numbers or strings when defining concepts. For example, the concept description *Human* $\sqcap \geq_{18}(\mathit{age})$ describes adult human beings. However, only very restricted forms of concrete domains are admissible in $\mathcal{EL}^{++}$ (see [5] for details).
- *Restricted role-value maps* are of the form $r_1 \circ \ldots \circ r_k \sqsubseteq r$. They are TBox axioms and not concept constructors. In a model of this role-value map, the composition of the roles $r_1, \ldots, r_k$ must be contained in the role r. Special cases of such role-value maps are *transitivity of a role* r, expressed as $r \circ r \sqsubseteq r$ and *right-identity rules* $r \circ s \sqsubseteq r$, which are both important for medical ontologies. For example, we may want to say that the *part_of* relation is transitive, which can be expressed as *part_of* $\circ$ *part_of* $\sqsubseteq$ *part_of*, and that medical findings are inherited along *part_of*, expressed as *finding_at* $\circ$ *part_of* $\sqsubseteq$ *finding_at*. Given the second role-value maps together with GCIs stating that a finger is part of the hand, an injury of the finger is an injury found at the finger, and an injury of the hand is an injury found at the hand, we can then deduce that an injury of the finger is an injury of the hand.
- A *reflexivity* axiom for the role r states that this role is reflexive, i.e., every individual is related to itself w.r.t. this role. For example, in a medical ontology one may want to state that the *part_of* relation is reflexive, i.e., every entity is part of itself.
- The range restriction $\mathit{ran}(r) \sqsubseteq C$ says that the second component of every tuple belonging to r must belong to C. For example, the range restriction $\mathit{ran}(\mathit{finding\_at}) \sqsubseteq$ *Body_structure* says that finding sites must belong to the body structure, i.e., this role is used to specify where in the body something (e.g., an injury) is found. The range restriction $\mathit{ran}(r) \sqsubseteq C$ could of course be expressed using the GCI $\top \sqsubseteq \forall r.C$, but value restrictions $\forall r.C$ are not available in $\mathcal{EL}^{++}$. Thus, range restrictions can be seen as a restricted way of using value restrictions in $\mathcal{EL}^{++}$. Note, however, that the unrestricted use of value restrictions would destroy tractability.

Note that the original version of $\mathcal{EL}^{++}$ [5] did not have reflexive roles and range restrictions. They were added in the version introduced in [6], which is the version of $\mathcal{EL}^{++}$ that underlies the designated OWL EL profile of OWL 2. To keep tractability (even decidability), one must actually impose a syntactic restriction on $\mathcal{EL}^{++}$-TBoxes that prevents interactions between restricted role-value maps and range restrictions (see [6] for details). It should also be noted that basically all other additions of typical DL constructors to $\mathcal{EL}$ make subsumption w.r.t. TBoxes ExpTime-hard [5, 6].

3 The DL-Lite family of DLs

DL-Lite$_{core}$ is the basic member of the DL-Lite family [20]. Concept descriptions of this DL are of the form

$$A, \ \exists r.\top, \ \exists r^-.\top$$

where A is a concept name, r is a role name, and r^- denotes the inverse of the role name r, with the obvious semantics

$$(r^-)^\mathcal{I} = \{(y, x) \mid (x, y) \in r^\mathcal{I}\}.$$

A DL-Lite$_{core}$ knowledge base (KB) consists of a TBox and an ABox. The *TBox formalism* allows for GCIs and disjointness axioms between DL-Lite$_{core}$ concept descriptions C, D:

$$C \sqsubseteq D \quad \text{and} \quad \mathsf{disj}(C, D),$$

where an interpretation $\mathcal{I}$ is a model of $\mathsf{disj}(C, D)$ if it satisfies $C^\mathcal{I} \cap D^\mathcal{I} = \emptyset$. Although conjunction is not available in DL-Lite$_{core}$, it can be simulated to a certain extent: a conjunction on the right-hand side of a GCI $C \sqsubseteq D_1 \sqcap D_2$ can be expressed by the two GCIs $C \sqsubseteq D_1$ and $C \sqsubseteq D_2$. Disjunction on the left-hand side of a GCI can be expressed in a similar way. The following is an example of a DL-Lite$_{core}$-TBox:

$$\begin{aligned}\mathcal{T}_{ex} = \{\exists \mathit{child}.\top \sqsubseteq \mathit{Parent}, &\quad \mathit{Parent} \sqsubseteq \mathit{Human},\\ \mathit{Human} \sqsubseteq \exists \mathit{child}^-.\top, &\quad \mathsf{disj}(\mathit{Human}, \mathit{Insect})\}.\end{aligned}$$

A DL-Lite$_{core}$-ABox is a finite set of *concept and role assertions*: $A(a)$ and $r(a, b)$, where A is a concept name, r is a role name, and a, b are individual names. An interpretation $\mathcal{I}$ assigns an element $c^\mathcal{I} \in \Delta^\mathcal{I}$ to every individual name c such that the *unique name assumption* (UNA) is satisfied, i.e. $a^\mathcal{I} \neq b^\mathcal{I}$ for distinct individual names a, b.[7] It is a model of $A(a)$ if it satisfies $a^\mathcal{I} \in$

[6]http://cel.googlecode.com

[7]The impact of dropping the UNA on the complexity of reasoning in the DL-Lite family has been investigated in [3].

$A^\mathcal{I}$ and of $r(a,b)$ if it satisfies $(a^\mathcal{I}, b^\mathcal{I}) \in r^\mathcal{I}$. The restriction to concept names in concept assertions can be circumvented by introducing a GCI for a new concept name, say $A_{new} \sqsubseteq C$, in the TBox and then stating $A_{new}(a)$ in the ABox. The following is an example of a DL-Lite$_{core}$-ABox:

$$\begin{aligned}\mathcal{A}_{ex} = \{ &\mathit{Woman}(\mathit{LINDA}), & &\mathit{child}(\mathit{LINDA}, \mathit{JAMES}),\\ &\mathit{Beatle}(\mathit{PAUL}), & &\mathit{child}(\mathit{PAUL}, \mathit{JAMES})\ \}.\end{aligned}$$

In [20], the following two extensions of DL-Lite$_{core}$ have also been considered:

- DL-Lite$_\mathcal{F}$, in which the TBox may additionally contain *functionality axioms* func(r) for role names and their inverses. Such an axiom can, e.g., be used to state that the role *father* is functional, i.e., every individual has at most one father.
- DL-Lite$_\mathcal{R}$, in which the TBox may additionally contain *role inclusion axioms* $r_1 \sqsubseteq r_2$ and *role disjointness axioms* disj(r_1, r_2) for role names and their inverses. Such axioms can, e.g., be used to state that the roles *father* and *mother* are disjoint subroles of $child^-$.

Other members of the DL-Lite family have, e.g., been defined in [21, 2, 37].

The DL-Lite family of DLs is tailored towards applications in which huge amounts of data (represented as an ABox) are queried w.r.t. fairly light-weight ontologies. In this setting, it is no longer sufficient that query answering is tractable. One needs to be able to store the ABox in a relational database system, and answer queries using a relational query engine. From a logical point of view, a relational database is a finite first-order interpretation $\mathcal{I}$, and the relational query engine can efficiently answer *first-order queries (FOL queries)*. Such a query is a first-order formula $\phi(\vec{x})$ over the vocabulary of the database and with free variables $\vec{x}$; an answer tuple $\vec{c}$ is a sequence of elements of the domain of $\mathcal{I}$ such that $\phi(\vec{c})$ evaluates to true in $\mathcal{I}$. Given an FOL query q, we denote the set of its answer tuples in the database $\mathcal{I}$ with $q^\mathcal{I}$.

In DL-Lite, one concentrates on answering a restricted form of FOL queries, so-called unions of conjunctive queries. A *conjunctive query* is a conjunction of atoms, built using concept and role names as predicate symbols, individual names as constant symbols, and variables, of which some may be existentially quantified. For example, the following is a conjunctive query:

$$\begin{aligned}q_{ex} = \exists y, z_1, z_2.\ &\mathit{Woman}(x) \wedge \mathit{child}(x,y) \wedge \mathit{child}(z_1,y) \wedge\\ &\mathit{Human}(z_1) \wedge \mathit{child}(z_2,z_1)\end{aligned}$$

A *union of conjunctive queries* is a finite set of conjunctive queries, which is interpreted as the disjunction of its elements. Given a union of conjunctive queries or a conjunctive query q and a knowledge base $\mathcal{K}$, the *set of answers to* q *over* $\mathcal{K}$ (denoted $ans(q,\mathcal{K})$) consists of all tuples $\vec{a}$ of individual names appearing in the knowledge base such that $\vec{a}^\mathcal{I} \in q^\mathcal{I}$ for every model $\mathcal{I}$ of the knowledge base. For the knowledge base $\mathcal{K}_{ex} = (\mathcal{T}_{ex}, \mathcal{A}_{ex})$ of our example and the conjunctive query q_{ex}, it is easy to see that $ans(q_{ex}, \mathcal{K}_{ex}) = \{\mathit{LINDA}\}$.

The approach for query answering in DL-Lite using a relational database system proceeds as follows:

1. use the TBox $\mathcal{T}$ to reformulate the given union of conjunctive queries q into an FOL query $q_\mathcal{T}$ and then discard the TBox;
2. view the ABox $\mathcal{A}$ as a relational database $\mathcal{I}_\mathcal{A}$, which has as its domain all individuals names occurring in $\mathcal{A}$, interprets concept names A as $A^{\mathcal{I}_\mathcal{A}} = \{a \mid A(a) \in \mathcal{A}\}$, and role names r as $r^{\mathcal{I}_\mathcal{A}} = \{(a,b) \mid r(a,b) \in \mathcal{A}\}$;
3. evaluate $q_\mathcal{T}$ in the database $\mathcal{I}_\mathcal{A}$ using a relational query engine.

If this approach is correct for a given DL $\mathcal{L}$, i.e., there is a reformulation function $q \mapsto q_\mathcal{T}$ such that ${q_\mathcal{T}}^{\mathcal{I}_\mathcal{A}} = ans(q, (\mathcal{T}, \mathcal{A}))$ for all unions of conjunctive queries q, then one says that answering conjunctive queries in $\mathcal{L}$ is *FOL-reducible.* The following theorem is proved in [20].

Theorem 2 *Answering conjunctive queries in DL-Lite$_{core}$, DL-Lite$_\mathcal{F}$, and DL-Lite$_\mathcal{R}$ is FOL-reducible.*

Since the size of the reformulated query does not depend on the size of the ABox, the data complexity of evaluating the original query (i.e., the complexity in terms of the size of the ABox) is the same as evaluating the reformulated query. Because the data complexity of evaluating FOL queries in a relational database is complete for the complexity class AC^0, this implies that the data complexity of answering conjunctive queries in DL-Lite$_{core}$, DL-Lite$_\mathcal{F}$, and DL-Lite$_\mathcal{R}$ is in AC^0, which is a proper subclass of the class of all tractable problems P. This method for query answering in DL-Lite based on FOL-reducibility has been implemented in the QuOnto system [1].

The reformulation approach developed in [20] actually yields a union of conjunctive queries rather than an arbitrary FOL query. Instead of describing it in detail, we illustrate it with our example. The main idea is to use the GCIs in the TBox as rewrite rules from right to left. Each rewrite step replaces an atom in a conjunctive query q contained in the union of conjunctive queries. The rewritten conjunctive query q' is then added to the union of conjunctive queries (without removing the original query q). Consider the atom $\mathit{child}(z_2, z_1)$ in q_{ex}. Since z_2 is existentially quantified, this basically says that z_1 belongs to $\exists \mathit{child}^-.\top$, and thus the GCI $\mathit{Human} \sqsubseteq \exists \mathit{child}^-.\top$ can be used to replace this atom with $\mathit{Human}(z_1)$, which already occurs in the conjunctive query. Thus, the new conjunctive query $q^{(1)}$:

$$\exists y, z_1. \mathit{Woman}(x) \wedge \mathit{child}(x,y) \wedge \mathit{child}(z_1,y)\ \wedge \mathit{Human}(z_1)$$

is added. In $q^{(1)}$, the atom $\mathit{Human}(z_1)$ can be replaced by $\mathit{Parent}(z_1)$, which yields the additional conjunctive query $q^{(2)}$. Using the GCI $\exists \mathit{child}.\top \sqsubseteq \mathit{Parent}$, the atom $\mathit{Parent}(z_1)$ in $q^{(2)}$ can be replaced by $\mathit{child}(z_1, z_3)$, where z_3 is a new existentially quantified variable. This yields the new conjunctive query $q^{(3)}$:

$$\exists y, z_1, z_3. \mathit{Woman}(x) \wedge \mathit{child}(x,y) \wedge \mathit{child}(z_1,y)\ \wedge \mathit{child}(z_1,z_3)$$

It is easy to see that *LINDA* is an answer for the query $q^{(3)}$ in the database $\mathcal{I}_{\mathcal{A}_{ex}}$, and thus of the union of conjunctive queries generated by the reformulation process. In addition to rewriting atoms using GCIs, the general reformulation process also uses unification of atoms in a conjunctive query to generate new conjunctive queries (see [20] for details).

It should be noted that also for (a fragment of) $\mathcal{EL}^{++}$, an approach to conjunctive query answering using relational database systems has been developed [40, 41]. Since the data complexity of query answering in $\mathcal{EL}$ is PTime-complete, the approach follows a different route than the one for DL-Lite (since FOL-reducibility implies that the data complexity of query answering is in AC^0). In particular, the TBox is incorporated into the ABox and not into the query. However, some limited query reformulation (independent of both the TBox and the ABox) is still required. Interestingly, both the ABox rewriting and the query reformulation cause only a polynomial blow-up, in contrast to DL-Lite, where the blow-up of the query may be exponential in the size of the original query [20]. This alternative approach for query answering using a relational database system can also be applied to DL-Lite [37]. The approach introduced in [37] causes an exponential blow-up of the query, but we believe that this may be avoidable. Nevertheless, even with this blow-up the query execution times are typically smaller than those of the approach introduced in [20].

4 Conclusion

We have described the origins of two novel families of lightweight DLs: logics of the $\mathcal{EL}$ family were designed to admit subsumption and classification in polynomial time, while still providing sufficient expressive power for life-science ontologies; logics of the DL-Lite family have been designed to enable query answering using relational database systems, while still providing sufficient expressive power to capture conceptual modelling formalisms. The relevance of the small DLs discussed in this article is underlined by the fact that both of them are captured in the official W3C profiles[8] document for the candidate recommendation of OWL 2. Each of the OWL 2 profiles are designed for specific application requirements. For applications that rely on reasoning services for ontologies with a large number of concepts, the profile OWL 2 EL has been introduced, which is based on $\mathcal{EL}^{++}$. For applications that deal with large sets of data and that mainly use the reasoning service of query answering, the profile OWL 2 QL has been defined. The DL underlying this profile is DL-Lite$_{\mathcal{R}}$. Both, the profile OWL 2 EL and OWL 2 QL pave the way to apply very efficient reasoning services in practical applications. The recent research and standardization efforts discussed in this paper suggest that small is indeed again beautiful in Description Logics.

References

[1] A. Acciarri, D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QUONTO: QUerying ONTOlogies. In *Proc. of the Nat. Conf. on AI (AAAI'05)*, 2005.

[2] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. DL-Lite in the light of first-order logic. In *Proc. of the Nat. Conf. on AI (AAAI'07)*, 2007.

[3] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. DL-Lite without the unique name assumption. In *Proc. of the Description Logic WS (DL'09), CEUR*, 2009.

[4] F. Baader. Terminological cycles in KL-ONE-based knowledge representation languages. In *Proc. of the Nat. Conf. on AI (AAAI'90)*, 1990.

[5] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the Int. Joint Conf. on AI (IJCAI'05)*, 2005.

[6] F. Baader, S. Brandt, and C. Lutz. Pushing the $\mathcal{EL}$ envelope further. In *Proc. of the Int. WS on OWL: Experiences and Directions (OWLED'08)*, 2008.

[7] F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *AIJ*, 88(1–2), 1996

[8] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider, eds. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003. Cambridge Univ. Press.

[9] F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. *Applied AI. Spec. Iss. on KB Management*, 1994.

[10] F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the Int. Joint Conf. on AI (IJCAI'91)*, 1991.

[11] F. Baader, I. Horrocks, and U. Sattler. Description logics. In *Handbook on Ontologies*, Int. Handbooks in Information Systems, 2003. Springer.

[12] F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the Int. Joint Conf. on AI (IJCAI'99)*, 1999.

[13] F. Baader, C. Lutz, and B. Suntisrivaraporn. Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In *Proc. of the Int. WS on Methods for Modalities (M4M-05)*, 2005.

[14] F. Baader, C. Lutz, and B. Suntisrivaraporn. CEL—a polynomial-time reasoner for life science ontologies. In *Proc. of the Int. Joint Conf. on Autom. Reasoning (IJCAR'06)*, LNAI 4130, 2006.

[15] F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69:5–40, 2001.

[16] R.J. Brachman and J.G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2),1985.

[17] S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and—what else? In *Proc. of the Eur. Conf. on AI (ECAI'04)*, 2004.

[18] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the Nat. Conf. on AI (AAAI'05)*, 2005.

[19] D. Calvanese, G. de Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Data complexity of query answering in description logics. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, 2006.

[20] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Autom. Reasoning*, 39(3), 2007.

[21] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, and R. Rosati. Linking data to ontologies: The description logic DL-Lite$_{\mathcal{A}}$. In *Proc. of the Int. WS on OWL: Experiences and Directions (OWLED'06), CEUR*, 2006.

[22] M. Fitting. Tableau methods of proof for modal logics. *Notre Dame J. of Formal Logic*, 13(2), 1972.

[23] V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In *Proc. of the Int. Joint Conf. on AI (IJCAI'01)*, 2001.

[24] V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Autom. Reasoning (IJCAR'01)*, LNAI 2083, 2001.

[25] V. Haarslev and R. Möller. On the scalability of description logic instance retrieval. *J. of Autom. Reasoning*, 41(2), 2008.

[26] B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, 1991.

[27] B. Hollunder, W. Nutt, and M. Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of the Eur. Conf. on AI (ECAI'90)*, 1990.

[8] `http://www.w3.org/TR/owl2-profiles/`

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

[29] I. Horrocks. Implementation and optimization techniques. In *[8]*, 2003.

[30] I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, 2006.

[31] I. Horrocks and P.F. Patel-Schneider. Optimizing description logic subsumption. *J. of Logic and Computation*, 9(3),1999.

[32] I. Horrocks and P.F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *J. Web Semantics*, 1(4), 2004.

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

[34] I. Horrocks and U. Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the Int. Joint Conf. on AI (IJCAI'05)*, 2005.

[35] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *J. of the Interest Group in Pure and Applied Logic*, 8(3), 2000.

[36] Y. Kazakov. $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In *Proc. of the Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'08)*, 2008.

[37] R. Kontchakov, C. Lutz, D. Toman, F. Wolter, and M. Zakharyaschev. Combined FO rewritability for conjunctive query answering in DL-Lite. In *Proc. of the Description Logic WS (DL'09)*, 2009.

[38] H.J. Levesque and R.J. Brachman. Expressiveness and tractability in knowledge representation and reasoning. *Computational Intelligence*, 3, 1987.

[39] C. Lutz. The complexity of conjunctive query answering in expressive description logics. In *Proc. of the Int. Joint Conf. on Autom. Reasoning (IJCAR'08)*, LNAI 5195, 2008.

[40] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in $\mathcal{EL}$ using a database system. In *In Proc. of the Int. WS on OWL: Experiences and Directions (OWLED'08)*, 2008.

[41] C. Lutz, D. Toman, and F. Wolter. Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In *Proc. of the Int. Joint Conf. on AI (IJCAI'09)*, 2009.

[42] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In *Principles of Semantic Networks*, 1991. Morgan Kaufmann.

[43] E. Mays, R. Dionne, and R. Weida. K-REP system overview. *SIGART Bull.*, 2(3), 1991.

[44] B. Nebel. Computational complexity of terminological reasoning in BACK. *AIJ*, 34(3), 1988.

[45] B. Nebel. Terminological reasoning is inherently intractable. *AIJ*, 43(2), 1990.

[46] M. Ortiz, D. Calvanese, and T. Eiter. Data complexity of query answering in expressive description logics via tableaux. *J. of Autom. Reasoning*, 41(1), 2008.

[47] P.F. Patel-Schneider. Small can be beautiful in knowledge representation. In *Proc. of the IEEE WS on Knowledge-Based Systems*, 1984.

[48] Ch. Peltason. The BACK system — an overview. *SIGART Bull.*, 2(3), 1991.

[49] M. Schmidt-Schauß. Subsumption in KL-ONE is undecidable. In *Proc. of the Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'89)*, 1989.

[50] M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *AIJ*, 48(1), 1991.

[51] E. Sirin and B. Parsia. Pellet: An OWL DL reasoner. In *Proc. of the Description Logic WS (DL'04)*, 2004.

[52] B. Suntisrivaraporn. *Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies.* PhD thesis, Fakultät Informatik, TU Dresden, 2009.

[53] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Autom. Reasoning (IJCAR'06)*, LNAI 4130, 2006.

Contact

Franz Baader, Anni-Yasmin Turhan
Institut für Theoretische Informatik
TU Dresden
01062 Dresden
Email: [baader | turhan]@inf.tu-dresden.de

Carsten Lutz
Universität Bremen
Fachbereich 03
Postfach 330440
28334 Bremen
Email: clu@informatik.uni-bremen.de

Bild

Franz Baader is director of the Institute for Theoretical Computer Science at TU Dresden. His main research areas are knowledge representation (in particular description and modal logics) and automated deduction (in particular term rewriting, unification, and combination of constraint solvers).

Bild

Carsten Lutz is professor of computer science at the University of Bremen. His research interests mainly concern applications of modal logic in computer science, including description logics, database formalisms such as XPath, and logics for automated verification.

Bild

Anni-Yasmin Turhan is teaching and research fellow at TU Dresden. Her research interests include reasoning in Description Logics, in particular non-standard inferences in DLs and implementations of DL reasoning systems.