What's new in Description Logics

Franz Baader

Mainstream DL research of the last 25 years: towards very expressive DLs with practical inference procedures

Description Logics [5] 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 (http://www.w3.org/TR/owl-features/) as the standard ontology language for the Semantic Web [37].

The name *Description Logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by *concept descriptions*, that is expressions that are built from atomic concepts (unary predicates) and atomic roles (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 names *Man*, *Doctor*, and *Happy* and the role names *married* and *child*, the concept of "a man that is married to a doctor, and has only happy children" can be expressed using the concept description

$$Man \sqcap \exists married.\, Doctor \sqcap \forall child.\, Happy.$$

On the other hand, DLs differ from their predecessors in that they are equipped with a formal, *logic*-based semantics, which can, for example be given by a translation into first-order predicate logic. For example, the above concept description can be translated into the following first-order formula (with one free variable x):

$$Man(x) \wedge \exists y.\, (married(x,y) \wedge \mathsf{Doctor}(y)) \\ \wedge \forall y.\, (child(x,y) \rightarrow Happy(y)).$$

The motivation for introducing the early predecessors of DLs, such as semantic networks and frames [50, 57], actually was to develop means of representation that are closer to the way humans represent knowledge than a representation in formal logics, like first-order predicate logic. Minsky [50] even combined his introduction of the frame idea with a general rejection of logic as an appropriate formalism for representing knowledge. However, once people tried to equip these "formalisms" with a formal semantics, it turned out that they can be seen as syntactic variants of (subclasses of) first-order predicate logic [33, 63]. Description Logics were developed with the intention of keeping the advantages of the logic-based approach to knowledge representation (like a formal model-theoretic semantics and well-defined inference problems), while avoiding the disadvantages of using full first-order predicate logic (e. g., by using a variable-free syntax that is easier to read, and by ensuring decidability of the important inference problems).

DOI 10.1007/s00287-011-0534-y

Franz Baader
Theoretical Computer Science, TU Dresden, Dresden
E-Mail: baader@tcs.inf.tu-dresden.de

Abstract

Main stream research in Description Logics (DLs) until recently concentrated on *increasing the expressive power* of the employed description language while keeping *standard inference problems* like subsumption and instance manageable in the sense that highly-optimized reasoning procedure for them behave well in practice. One of the main successes of this line of research was the adoption of OWL DL, which is based on an expressive DL, as the standard ontology language for the Semantic Web.

More recently, there has been a growing interest in more *light-weight DLs*, and in *other kinds of inference problems*, mainly triggered by need in applications with large-scale ontologies. In this paper, we first review the DL research leading to the very expressive DLs with practical inference procedures underlying OWL, and then sketch the recent development of light-weight DLs and novel inference procedures.

Concept descriptions can be used to define the terminology of the application domain, and to make statements about a specific application situation in the assertional part of the knowledge base. 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*

$Man \equiv Human \sqcap \neg Female$,

$Woman \equiv Human \sqcap Female$,

$Father \equiv Man \sqcap \exists child.\top$

define the concept of a man (woman) as a human that is not female (is female), and the concept of a father as a man that has a child, where $\top$ stands for the top concept (which is interpreted as the universe of all individuals in the application domain). The above is a (very simple) example of an *acyclic TBox*, which is a finite set of concept definitions that is unambiguous (i. e., every concept name appears at most once on the left-hand side of a definition) and acyclic (i. e., there are no cyclic dependencies between definitions). In *general TBoxes*, 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

$\exists child.Human \sqsubseteq Human$ and

$Human \sqsubseteq \forall child.Human$

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

$Man(JOHN)$, $child(JOHN, MACKENZIE)$,

$Female(MACKENZIE)$,

state that John is a man, who has the female child Mackenzie.

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* algorithm allows one to determine subconcept-superconcept relationships. For example, w.r.t. the concept definitions from above, the concept *Human* subsumes the concept *Father* since all instances of the second concept are necessarily instances of the first concept, that is whenever the above concept definitions are satisfied, then *Father* is interpreted as a subset of *Human*. With the help of the subsumption algorithm, one can compute the hierarchy of all concepts defined in a TBox. This inference service is usually called *classification*. The *instance* algorithm 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 *MACKENZIE* is an instance of the concept *Human*. With the help of the instance algorithm, one can compute answers to *instance queries*, that is all individuals occurring in the ABox that are instances of the query concept C.

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 specified using concept descriptions. 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, that is can be performed in polynomial time [55]. The precursor of all DL systems, KL-ONE [22], as well as its early successor systems, like Kandor [55], K-Rep [49], and Back [56], indeed employed polynomial-time subsumption algorithms. Later on, however, it turned out that subsumption in rather inexpressive DLs may be intractable [43], that subsumption in KL-ONE is even undecidable [61], 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 [51]. 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, that is 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 [52] 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.[1]

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 [62], though it had already been used for modal logics long before that [29]. It has turned out that this approach can be used to handle a great variety of different DLs (see [18] for an overview and, e. g., [35, 38, 45] for more recent results), 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 [6, 32, 34], even for expressive DLs with a high worst-case complexity (ExpTime and beyond). The advent of 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++ [67], Racer [31], and Pellet [64] 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 in the worst-case NExpTime-complete [36].

The research on how to extend the expressive power of DLs has actually not stopped with the adoption of $\mathcal{SHOIN}(\mathcal{D})$ as the DL underlying OWL. In fact, the new version of the OWL standard, OWL 2 (http://www.w3.org/TR/2009/REC-owl2-overview-20091027/), is based on the even more expressive DL $\mathcal{SROIQ}(\mathcal{D})$, which is 2NExpTime-complete [40]. The main new features of $\mathcal{SROIQ}(\mathcal{D})$ are the use of

[1] 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 $\mathcal{FL}_0$ [2].

qualified number restrictions ($\mathcal{Q}$) rather than simple number restrictions ($\mathcal{N}$), and the availability of (a restricted form of) role inclusion axioms ($\mathcal{R}$). For example, with a simple number restriction we can describe the concept of a man that has three children

$$Man \sqcap (\geq 3\, child),$$

but we cannot specify properties of these children, as in the qualified number restriction

$$Man \sqcap (\geq 3\, child.\, Happy).$$

More recent developments: Light-weight DLs and the need for novel inference tools

In this section, we first discuss the $\mathcal{EL}$ and the DL-Lite families of light-weight DLs, and then consider inference problems different from the subsumption and the instance problem.

Light-weight DLs: The $\mathcal{EL}$ family

The ever increasing expressive power and worst-case complexity of expressive DLs, combined with the increased use of DL-based ontology languages in practical applications due to the OWL standard, has also resulted in an increasing number of ontologies 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 (http://www.ihtsdo.org/snomed-ct/), which comprises 380.000 concepts and is used as a standardized health care terminology 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 [12],[2] and Pellet still could not classify SNOMED CT in tests performed in 2008 [66].

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 $\mathcal{EL}$ [11]. 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 $\mathcal{EL}$ [11] and $\mathcal{FL}_0$ [43], subsumption in $\mathcal{FL}_0$ w.r.t. an acyclic TBox is coNP-complete [52] and w.r.t. GCIs it is even ExpTime-complete [3]. In contrast, subsumption in $\mathcal{EL}$ stays tractable even w.r.t. GCIs [23], and this result is stable under the addition of several interesting means of expressivity [3, 4].

The *polynomial-time subsumption algorithm* for $\mathcal{EL}$ [3, 23] actually classifies the given TBox $\mathcal{T}$, that is 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 [23]. In the next step, a *classification graph* $G_{\mathcal{T}} = (V, V \times V, S, R)$ is built, 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*:

- $S(A)$ contains only subsumers of A w.r.t. $\mathcal{T}$.
- $R(A, B)$ 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 Fig. 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. The fact that subsumption in $\mathcal{EL}$ w.r.t. TBoxes can be

[2] Note, however, that more recent versions of FaCT++ and Racer perform quite well on SNOMED CT [66], due to optimizations specifically tailored towards the classification of SNOMED CT.

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

Fig. 1 The completion rules for subsumption in $\mathcal{EL}$ w.r.t. general TBoxes

decided in polynomial time is an immediate consequence of the facts that 1) rule application terminates after a polynomial number of steps, and 2) if no more rules are applicable then $S(A)$ contains exactly those concept names B occurring in $\mathcal{T}$ that are subsumers of A w.r.t. $\mathcal{T}$ (see [3, 23] for more details and full proofs).

Light-weight DLs: The DL-Lite family

Another problematic issue with expressive DLs is that query answering in such DLs does not scale too well to knowledge bases with a very large ABox. In this context, *queries* are conjunctions of assertions that may also contain variables, of which some can be existentially quantified. For example, the query

$$\exists y.\, Man(x) \wedge child(x,y) \wedge Woman(y)$$

asks for all men that have a child that is a woman.[3] In the database world, these kinds of queries are called *conjunctive queries* [1]; the difference to the pure database case is that, in addition to the instance data, we also have a TBox. As an example, consider the ABox assertions stating facts about John and Mackenzie from the previous section. Without any additional information about the meaning of the predicates *Man*, *child*, and *Woman*, the individual *JOHN* is not an answer to the above query. However, if we take the concept definitions and GCIs introduced in the previous section into account, then *JOHN* turns out to be an answer to this query.

Query answering in expressive DLs such as the already mentioned $\mathcal{SHOIN}$ (i. e., $\mathcal{SHOIN}(\mathcal{D})$ without concrete domains) is 2ExpTime-complete regarding combined complexity [44], that is the complexity w.r.t. the size of the TBox and the ABox. Thus, query answering in this logic is even harder than subsumption while at the same time being much more time critical. Moreover, query answering in $\mathcal{SHOIN}$ is coNP-complete [53] regarding data complexity (i. e., in the size of the ABox), which is viewed as "unfeasible" in the database community. These complexity hardness results for answering conjunctive queries in expressive DLs are dramatic since many DL applications, such as those that use ABoxes as 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 [24–26], designed to allow the implementation of conjunctive query answering "on top of" a relational database system.

DL-Litecore is the basic member of the DL-Lite family [26]. Concept descriptions of this DL are of the form

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

where A is a concept name, r is a role name, and r^- denotes the inverse of the role name r. A DL-Litecore knowledge base (KB) consists of a TBox and an ABox. The *TBox formalism* allows for GCIs and disjointness axioms between DL-Litecore concept descriptions C, D:

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

where $\mathsf{disj}(C,D)$ states that C, D must always be interpreted as disjoint sets. A DL-Litecore-ABox is a finite set of *concept and role assertions*: $A(a)$ and

[3] This simple query could also be expressed as an instance query using the $\mathcal{EL}$-concept description $Man \sqcap \exists child.Woman$, but in general the use of variables allows the formulation of more complex queries than simple instance queries.

$r(a, b)$, where A is a concept name, r is a role name, and a, b are individual names.

In contrast to $\mathcal{EL}$, DL-Lite cannot express *qualified* existential restrictions such as $\exists child.Woman$ in the TBox. Conversely, $\mathcal{EL}$ does not have inverse roles, which are available (albeit in a limited way) in DL-Lite.

In principle, query answering in DL-Lite can be realized as follows:

1. use the TBox $\mathcal{T}$ to reformulate the given conjunctive queries q into a first-order query $q_{\mathcal{T}}$ and then discard the TBox,
2. view the ABox $\mathcal{A}$ as a relational database $\mathcal{I}_{\mathcal{A}}$,
3. evaluate $q_{\mathcal{T}}$ in the database $\mathcal{I}_{\mathcal{A}}$ using a relational query engine.

In practice, more work needs to be done to turn this into a scalable approach for query answering. For example, the queries $q_{\mathcal{T}}$ generated by the reformulation step are very different from the SQL queries usually formulated by humans, and thus relational database engines are not optimized for such queries.

Interestingly, also in $\mathcal{EL}$ it is possible to implement query answering using a relational database system [46]. In contrast to the approach for DL-Lite, the TBox is incorporated into the ABox and not into the query. In addition, some limited query reformulation (independent of both the TBox and the ABox) is also required.

The relevance of the light-weight DLs discussed above is underlined by the fact that both of them are captured in the official W3C profiles (http://www.w3.org/TR/owl2-profiles/) document for 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 $\mathcal{EL}^{++}$, a tractable extension of $\mathcal{EL}$. 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 a member of the DL-Lite family.

Novel inference problems

The developers of the early DL systems concentrated on the subsumption and the instance problem, and the same was true until recently for the developers of highly optimized systems for expressive DLs. The development, maintenance, and usage of large ontologies can, however, also profit from the use of other inference procedures. Certain nonstandard inference problems, like *unification* [13, 14], *matching* [8, 10], and the problem of computing *least common subsumers* [7, 11, 19, 28] have been investigated for quite a while [9]. Unification and matching can, for example, help the ontology engineer to find redundancies in large ontologies, and least common subsumers and most specific concepts can be used to generate concepts from examples.

Other nonstandard inference problems have, however, come into the focus of mainstream DL research only recently. One example is *conjunctive query answering*, which is not only investigated for light-weight DLs (see above), but also for expressive DLs [30, 44].

Another is identification and extraction of *modules* inside an ontology. Intuitively, given an ontology $\mathcal{O}$ and a signature Σ (i. e., a subset of the concept and role names occurring in $\mathcal{O}$), a module $\mathcal{M}$ is a subset of $\mathcal{O}$ such that the following holds for all concept descriptions C, D that can be built from symbols in Σ: C is subsumed by D w.r.t. $\mathcal{O}$ iff C is subsumed by D w.r.t. $\mathcal{M}$. Consequently, if one is only interested in subsumption between concepts built from symbols in Σ, it is sufficient to use $\mathcal{M}$ instead of the (possibly much larger) whole ontology $\mathcal{O}$. Similarly, one can also introduce the notion of a module for other inference problems (such as query answering). An overview over different approaches for defining modules and a guideline for when to use which notion of a module can be found in [59]. Module identification and extraction is computationally costly for expressive DLs, and even undecidable for very expressive ones such as OWL DL [47]. Both for the $\mathcal{EL}$ family [48, 65] and the DL-Lite family [41], the reasoning problems that are relevant in this area are decidable and usually of much lower complexity than for expressive DLs.

For a developer or user of a DL-based ontology, it is often quite hard to understand why a certain consequence computed by the reasoner actually follows from the knowledge base. For example, in the DL version of the medical ontology SNOMED CT, the concept *Amputation-of-Finger* is classified as a subconcept of *Amputation-of-Arm*. Finding the six axioms that are responsible for this error [20] among the more than 350 000 concept definitions of SNOMED CT without support by an automated

reasoning tool is not easy. *Axiom pinpointing* [60] has been introduced to help developers or users of DL-based ontologies understand the reasons why a certain consequence holds by computing minimal subsets of the knowledge base that have the consequence in question (called MinAs or Explanations). There are two general approaches for computing MinAs: the *black-box* approach and the *glass-box* approach. The most naïve variant of the black-box approach considers all subsets of the ontology, and computes for each of them whether it still has the consequence or not. More sophisticated versions [39] use a variant of Reiter's [58] hitting set tree algorithm to compute all MinAs. Instead of applying such a black-box approach to a large ontology, one can also first try to find a small and easy to compute subset of the ontology that contains all MinAs, and then apply the black-box approach to this subset [20]. The main advantage of the black-box approach is that it can use existing highly optimized DL reasoners unchanged. However, it may be necessary to call the reasoner an exponential number of times. In contrast, the glass-box approach tries to find all MinAs by a single run of a modified reasoner.

Most of the glass-box pinpointing algorithms described in the DL literature (e. g., [42, 54, 60]) are obtained as extensions of tableau-based reasoning algorithms [18] for computing consequences from DL knowledge bases. To overcome the problem of having to design a new pinpointing extension for every tableau-based algorithm, the papers [15, 17] introduce a general approach for extending tableau-based algorithms to pinpointing algorithms. This approach is based on a general notion of "tableau algorithm," which captures many of the known tableau-based algorithms for DLs and Modal Logics, but also other kinds of decision procedures, like the polynomial-time subsumption algorithm for the DL $\mathcal{EL}$ sketched above. Any such tableau algorithm can be extended to a pinpointing algorithm, which is correct in the sense that a terminating run of the algorithm computes all MinAs. Unfortunately, however, termination need not transfer from a given tableau to its pinpointing extension, and the approach only applies to tableau-based algorithms that terminate without requiring any cycle-checking mechanism (usually called "blocking" in the DL community). Though these problems can, in principle, be solved by restricting the general framework to so-called forest tableaux [17], this solution makes the definitions and proofs more complicated and less intuitive.

In [16], a different general approach for obtaining glass-box pinpointing algorithms, which also applies to DLs for which the termination of tableau-based algorithms requires the use of blocking, is presented. It is well-known that automata working on infinite trees can often be used to construct worst-case optimal decision procedures for such DLs [21, 27]. In this automata-based approach, the input inference problem Γ is translated into a tree automaton $\mathcal{A}_\Gamma$, which is then tested for emptiness. Basically, pinpointing is then realized by transforming the tree automaton $\mathcal{A}_\Gamma$ into a weighted tree automaton working on infinite trees, and computing the so-called behavior of this weighted automaton.

Conclusion

The DL research of the last 30 years has led, on the one hand, to highly expressive ontology languages, which can nevertheless be supported by practical reasoning tools. On the other hand, the recent development of light-weight DLs and specialized reasoning tools for them ensures that DL reasoning scales to large ontologies with hundreds of thousands of terminological axioms (like SNOMED CT) and, by using database technology, to much larger sets of instance data. In addition, novel inference methods such as modularization and pinpointing support building and maintaining high-quality ontologies.

References

1. Abiteboul S, Hull R, Vianu V (1995) Foundations of Databases. Addison Wesley, Reading, MA
2. Baader F (1990) Terminological cycles in KL-ONE-based knowledge representation languages. In: Proc of the 8th Nat Conf on Artificial Intelligence (AAAI'90), 29 July–3 August 1990, Boston, MA, USA, pp 621–626
3. Baader F, Brandt S, Lutz C (2005) Pushing the $\mathcal{EL}$ envelope. In: Kaelbling LP, Saffiotti A (eds) Proc of the 19th Int Joint Conf on Artificial Intelligence (IJCAI 2005), 30 July–5 August 2005, Edinburgh, UK. Morgan Kaufmann, Los Altos, pp 364–369
4. Baader F, Brandt S, Lutz C (2008) Pushing the $\mathcal{EL}$ envelope further. In: Clark K, Patel-Schneider PF (eds) Proc of the Fifth International Workshop on OWL: Experiences and Directions (OWLED'08), 26–27 October 2008, Karlsruhe, Germany
5. Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) (2003) The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press
6. Baader F, Franconi E, Hollunder B, Nebel B, Profitlich H-J (1994) An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. Appl Artif Intel. Special Issue on Knowledge Base Management 4:109–132
7. Baader F, Küsters R (1998) Computing the least common subsumer and the most specific concept in the presence of cyclic $\mathcal{ALN}$-concept descriptions. In: Proc of the 22nd German Annual Conf on Artificial Intelligence (KI'98), vol 1504, Lecture Notes in Computer Science. Springer, pp 129–140

8. Baader F, Küsters R (2000) Matching in description logics with existential restrictions. In: Proc of the 7th Int Conf on Principles of Knowledge Representation and Reasoning (KR 2000), 12–15 April 2000, Breckenridge, CO, USA, pp 261–272
9. Baader F, Küsters R (2006) Nonstandard inferences in description logics: the story so far. In: Gabbay DM, Goncharov SS, Zakharyaschev M (eds) Mathematical Problems from Applied Logic I, vol 4, International Mathematical Series. Springer, pp 1–75
10. Baader F, Küsters R, Borgida A, McGuinness DL (1999) Matching in description logics. J Logic Comput 9(3):411–447
11. Baader F, Küsters R, Molitor R (1999) Computing least common subsumers in description logics with existential restrictions. In: Proc of the 16th Int Joint Conf on Artificial Intelligence (IJCAI'99), 31 July–6 August 1999, Stockholm, Sweden, pp 96–101
12. Baader F, Lutz C, Suntisrivaraporn B (2005) Is tractable reasoning in extensions of the description logic $\mathcal{EL}$ useful in practice? In: Proc of the 2005 International Workshop on Methods for Modalities (M4M-05), 1–2 December 2005, Berlin, Germany
13. Baader F, Morawska B (2009) Unification in the description logic $\mathcal{EL}$. In: Treinen R (ed) Proc of the 20th Int Conf on Rewriting Techniques and Applications (RTA 2009), vol 5595, Lecture Notes in Computer Science. Springer, pp 350–364
14. Baader F, Narendran P (2001) Unification of concepts terms in description logics. J Symb Comput 31(3):277–305
15. Baader F, Peñaloza R (2007) Axiom pinpointing in general tableaux. In: Proc of the Int Conf on Analytic Tableaux and Related Methods (TABLEAUX 2007), vol 4548, Lecture Notes in Artificial Intelligence. Springer, pp 11–27
16. Baader F, Peñaloza R (2008) Automata-based axiom pinpointing. In: Armando A, Baumgartner P, Dowek G (eds) Proc of the Int Joint Conf on Automated Reasoning (IJCAR 2008), vol 5195 Lecture Notes in Artificial Intelligence. Springer, pp 226–241
17. Baader F, Peñaloza R (2010) Axiom Pinpointing in General Tableaux. J Logic Comput 20(1):5–34
18. Baader F, Sattler U (2001) An overview of tableau algorithms for description logics. Studia Logica 69:5–40
19. Baader F, Sertkaya B, Turhan A-Y (2007) Computing the least common subsumer w.r.t a background terminology. J Appl Logic 5(3):392–420
20. Baader F, Suntisrivaraporn B (2008) Debugging SNOMED CT using axiom pinpointing in the description logic $\mathcal{EL}^+$. In: Proc of the International Conference on Representing and Sharing Knowledge Using SNOMED (KR-MED'08), 31 May–2 June 2008, Phoenix, Arizona
21. Baader F, Tobies S (2001) The inverse method implements the automata approach for modal satisfiability. In: Proc of the Int Joint Conf on Automated Reasoning (IJCAR 2001), vol 2083, Lecture Notes in Artificial Intelligence. Springer, pp 92–106
22. Brachman RJ, Schmolze JG (1985) An overview of the KL-ONE knowledge representation system. Cognitive Sci 9(2):171–216
23. Brandt S (2004) Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else? In: López de Mántaras R, Saitta L (eds) Proc of the 16th Eur Conf on Artificial Intelligence (ECAI 2004), 22–27 August 2004, Valencia, Spain, pp 298–302
24. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2005) DL-Lite: Tractable description logics for ontologies. In: Veloso MM, Kambhampati S (eds) Proc of the 20th Nat Conf on Artificial Intelligence (AAAI 2005), 25–29 July 2004, San Jose, CA, USA. AAAI Press/The MIT Press, pp 602–607
25. Calvanese D, de Giacomo G, Lembo D, Lenzerini M, Rosati R (2006) Data complexity of query answering in description logics. In: Doherty P, Mylopoulos J, Welty CA (eds) Proc of the 10th Int Conf on Principles of Knowledge Representation and Reasoning (KR 2006), 2–5 June 2006, Lake District, UK. AAAI Press/The MIT Press, pp 260–270
26. Calvanese D, De Giacomo G, Lembo D, Lenzerini M, Rosati R (2007) Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J Autom Reason 39(3):385–429
27. Calvanese D, De Giacomo G, Lenzerini M (2002) 2ATAs make DLs easy. In: Proc of the 2002 Description Logic Workshop (DL 2002). CEUR Electronic Workshop Proceedings. http://ceur-ws.org/Vol-53/, last access 5.4.2011, pp 107–118
28. Donini FM, Colucci S, Di Noia T, Di Sciascio E (2009) A tableaux-based method for computing least common subsumers for expressive description logics. In: Boutilier C (ed) Proc of the 21st Int Joint Conf on Artificial Intelligence (IJCAI 2009), 11–17 July 2009, Pasadena, CA, USA, pp 739–745
29. Fitting M (1972) Tableau methods of proof for modal logics. Notre Dame J Formal Logic 13(2):237–247
30. Glimm B, Horrocks I, Lutz C, Sattler U (2007) Conjunctive query answering for the description logic $\mathcal{SHIQ}$. In: Veloso MM (ed) Proc of the 20th Int Joint Conf on Artificial Intelligence (IJCAI 2007), 6–12 January 2007, Hyderabad, India, pp 399–404
31. Haarslev V, Möller R (2001) RACER system description. In: Proc of the Int Joint Conf on Automated Reasoning (IJCAR 2001), vol 2083, Lecture Notes in Artificial Intelligence. Springer, pp 701–706
32. Haarslev V, Möller R (2008) On the scalability of description logic instance retrieval. J Autom Reason 41(2):99–142
33. Hayes PJ (1979) The logic of frames. In: Metzing D (ed) Frame Conceptions and Text Understanding. Walter de Gruyter, pp 46–61 (republished in: Brachman RJ, Levesque HJ (eds) (1985) Readings in Knowledge Representation. Morgan Kaufmann, Los Altos)
34. Horrocks I (2003) Implementation and optimization techniques. In: Baader F, Calvanese D, McGuinness D, Nardi D, Patel-Schneider PF (eds) The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, pp 306–346
35. Horrocks I, Kutz O, Sattler U (2006) The even more irresistible $\mathcal{SROIQ}$. In: Doherty P, Mylopoulos J, Welty CA (eds) Proc of the 10th Int Conf on Principles of Knowledge Representation and Reasoning (KR 2006) 2–5 June 2006, Lake District, UK. AAAI Press/The MIT Press, pp 57–67
36. Horrocks I, Patel-Schneider PF (2004) Reducing OWL entailment to description logic satisfiability. J Web Sem 1(4):345–357
37. Horrocks I, Patel-Schneider PF, van Harmelen F (2003) From SHIQ and RDF to OWL: The making of a web ontology language. J Web Sem 1(1):7–26
38. Horrocks I, Sattler U (2005) A tableaux decision procedure for $\mathcal{SHOIQ}$. In: Proc of the 19th Int Joint Conf on Artificial Intelligence (IJCAI 2005), 30 July–5 August 2005, Edinburgh, UK. Morgan Kaufmann, Los Altos
39. Kalyanpur A, Parsia B, Horridge M, Sirin E (2007) Finding all justifications of OWL DL entailments. In: Proc of the 6th International Semantic Web Conference and 2nd Asian Semantic Web Conference, ISWC 2007+ASWC 2007, Busan, Korea, vol 4825, Lecture Notes in Computer Science. Springer, pp 267–280
40. Kazakov Y (2008) $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Brewka G, Lang J (eds) Proc of the 11th Int Conf on Principles of Knowledge Representation and Reasoning (KR 2008), 16–19 September 2008, Sydney, Australia. AAAI Press, pp 274–284
41. Kontchakov R, Wolter F, Zakharyaschev M (2008) Can you tell the difference between DL-Lite ontologies? In: Brewka G, Lang J (eds) Proc of the 11th Int Conf on Principles of Knowledge Representation and Reasoning (KR 2008), 16–19 September 2008, Sydney, Australia. Morgan Kaufmann, Los Altos, pp 285–295
42. Lee K, Meyer T, Pan JZ (2006) Computing maximally satisfiable terminologies for the description logic $\mathcal{ALC}$ with GCIs. In: Proc of the 2006 Description Logic Workshop (DL 2006), vol 189 CEUR Electronic Workshop Proceedings
43. Levesque HJ, Brachman RJ (1987) Expressiveness and tractability in knowledge representation and reasoning. Comput Intell 3:78–93
44. Lutz C (2008) The complexity of conjunctive query answering in expressive description logics. In: Armando A, Baumgartner P, Dowek G (eds) Proc of the Int Joint Conf on Automated Reasoning (IJCAR 2008), Lecture Notes in Artificial Intelligence. Springer, pp 179–193
45. Lutz C, Milicic M (2007) A tableau algorithm for description logics with concrete domains and general TBoxes. J Autom Reason 38(1–3):227–259
46. Lutz C, Toman D, Wolter F (2009) Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc of the 21st International Joint Conference on Artificial Intelligence IJCAI09, 11–17 July 2009, Pasadena, CA, USA. AAAI Press
47. Lutz C, Walther D, Wolter F (2007) Conservative extensions in expressive description logics. In: Veloso MM (ed) Proc of the 20th Int Joint Conf on Artificial Intelligence (IJCAI 2007), 6–12 January 2007, Hyderabad, India, pp 453–458
48. Lutz C, Wolter F (2007) Conservative extensions in the lightweight description logic $\mathcal{EL}$. In: Pfenning F (ed) Proc of the 21st Int Conf on Automated Deduction (CADE 2007), vol 4603, Lecture Notes in Computer Science, Bremen, Germany. Springer, pp 84–99
49. Mays E, Dionne R, Weida R (1991) K-REP system overview. SIGART Bull 2(3): 93–97
50. Minsky M (1981) A framework for representing knowledge. In: Haugeland J (ed) Mind Design. The MIT Press. A longer version appeared in: The Psychology of Computer Vision (1975) (republished in: Brachman RJ, Levesque HJ (eds) (1985) Readings in Knowledge Representation. Morgan Kaufmann, Los Altos)
51. Nebel B (1988) Computational complexity of terminological reasoning in BACK. Artif Intell 34(3):371–383
52. Nebel B (1990) Terminological reasoning is inherently intractable. Artif Intell 43: 235–249
53. Ortiz M, Calvanese D, Eiter T (2008) Data complexity of query answering in expressive description logics via tableaux. J Autom Reason 41(1):61–98

54. Parsia B, Sirin E, Kalyanpur A (2005) Debugging OWL ontologies. In: Ellis A, Hagino T (eds) Proc of the 14th International Conference on World Wide Web (WWW'05), 10–14 May 2005, Chiba, Japan. ACM, pp 633–640
55. Patel-Schneider PF (1984) Small can be beautiful in knowledge representation. In: Proc of the IEEE Workshop on Knowledge-Based Systems, 2–4 December 1984, Denver, CO, USA. An extended version appeared as Fairchild Tech Rep 660 and FLAIR Tech Rep 37, October 1984
56. Peltason C (1991) The BACK system – an overview. SIGART Bull 2(3):114–119
57. Quillian MR (1967) Word concepts: a theory and simulation of some basic capabilities. Behavioral Sci 12:410–430 (republished in: Brachman RJ, Levesque HJ (eds) (1985) Readings in Knowledge Representation. Morgan Kaufmann, Los Altos)
58. Reiter R (1987) A theory of diagnosis from first principles. Artif Intell 32(1): 57–95
59. Sattler U, Schneider T, Zakharyaschev M (2009) Which kind of module should I extract? In: Proc of the 2008 Description Logic Workshop (DL 2009), vol 477, CEUR Workshop Proceedings
60. Schlobach S, Cornet R (2003) Non-standard reasoning services for the debugging of description logic terminologies. In: Gottlob G, Walsh T (eds) Proc of the 18th Int Joint Conf on Artificial Intelligence (IJCAI 2003), Acapulco, Mexico. Morgan Kaufmann, Los Altos, pp 355–362
61. Schmidt-Schauß M (1989) Subsumption in KL-ONE is undecidable. In: Brachman RJ, Levesque HJ, Reiter R (eds) Proc of the 1st Int Conf on the Principles of Knowledge Representation and Reasoning (KR'89), 15–18 May 1989, Toronto, Ontario, Canada. Morgan Kaufmann, Los Altos, pp 421–431
62. Schmidt-Schauß M, Smolka G (1991) Attributive concept descriptions with complements. Artif Intell 48(1):1–26
63. Schubert LK, Goebel RG, Cercone NJ (1979) The structure and organization of a semantic net for comprehension and inference. In: Findler NV (ed) Associative Networks: Representation and Use of Knowledge by Computers. Academic Press, pp 121–175
64. Sirin E, Parsia B (2004) Pellet: An OWL DL reasoner. In: Proc of the 2004 Description Logic Workshop (DL 2004), 6–8 June 2004, Whistler, British Columbia, Canada, pp 212–213
65. Suntisrivaraporn B (2008) Module extraction and incremental classification: a pragmatic approach for $\mathcal{EL}^+$ ontologies. In: Bechhofer S, Hauswirth M, Hoffmann J, Koubarakis M (eds) Proc of the 5th European Semantic Web Conference (ESWC'08), vol 5021, Lecture Notes in Computer Science. Springer, pp 230–244
66. Suntisrivaraporn B (2009) Polynomial-Time Reasoning Support for Design and Maintenance of Large-Scale Biomedical Ontologies. PhD thesis, Fakultät Informatik, TU Dresden. http://lat.inf.tu-dresden.de/research/phd/#Sun-PhD-2008, last access 5.4.2011
67. Tsarkov D, Horrocks I (2006) Fact++ description logic reasoner: system description. In: Furbach U, Shankar N (eds) Proc of the Int Joint Conf on Automated Reasoning (IJCAR 2006), vol 4130, Lecture Notes in Artificial Intelligence. Springer, pp 292–297