Description Logics

Franz Baader[1], Ian Horrocks[2], and Ulrike Sattler[3]

[1] Institut für Theoretische Informatik, TU Dresden, Germany,
baader@tcs.inf.tu-dresden.de
[2] Computing Laboratory, Oxford University, Oxford, UK,
ian.horrocks@comlab.ox.ac.uk
[3] Department of Computer Science, University of Manchester, Manchester, UK,
sattler@cs.man.ac.uk

Summary. In this chapter, we explain what description logics are and why they make good ontology languages. In particular, we introduce the description logic $\mathcal{SHIQ}$, which has formed the basis of several well-known ontology languages, including OWL. We argue that, without the last decade of basic research in description logics, this family of knowledge representation languages could not have played such an important rôle in this context.

Description logic reasoning can be used both during the design phase, in order to improve the quality of ontologies, and in the deployment phase, in order to exploit the rich structure of ontologies and ontology based information. We discuss the extensions to $\mathcal{SHIQ}$ that are required for languages such as OWL and, finally, we sketch how novel reasoning services can support building ontologies.

1 Introduction

The aim of this section is to give a brief introduction to description logics, and to argue why they are well-suited as ontology languages. In the remainder of the chapter we will put some flesh on this skeleton by providing more technical details with respect to the theory of description logics, and their relationship to state of the art ontology languages. More detail on these and other matters related to description logics can be found in [6].

1.1 Ontologies

There have been many attempts to define what constitutes an ontology, perhaps the best known (at least amongst computer scientists) being due to Gruber: "an ontology is an explicit specification of a conceptualisation" [49].[1] In this context, a conceptualisation means an abstract model of some aspect

[1] This was later elaborated to "a formal specification of a shared conceptualisation" [21].

S. Staab and R. Studer (eds.), *Handbook on Ontologies,* International Handbooks on Information Systems, DOI 10.1007/978-3-540-92673-3,

of the world, taking the form of a definition of the properties of important concepts and relationships. An explicit specification means that the model should be specified in some unambiguous language, making it amenable to processing by machines as well as by humans.

Ontologies are becoming increasingly important in fields such as knowledge management, information integration, cooperative information systems, information retrieval and electronic commerce. One application area which has recently seen an explosion of interest is the so called *Semantic Web* [18], where ontologies are set to play a key rôle in establishing a common terminology between agents, thus ensuring that different agents have a shared understanding of terms used in semantic markup.

The effective use of ontologies requires not only a well-designed and well-defined ontology language, but also support from reasoning tools. Reasoning is important both to ensure the quality of an ontology, and in order to exploit the rich structure of ontologies and ontology based information. It can be employed in different phases of the ontology life cycle. During ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. In particular, one usually wants to compute the concept hierarchy, i.e. the partial ordering of named concepts based on the subsumption relationship. Information on which concept is a specialization of another, and which concepts are synonyms, can be used in the design phase to test whether the concept definitions in the ontology have the intended consequences or not. This information is also very useful when the ontology is deployed.

Since it is not reasonable to assume that all applications will use the same ontology, interoperability and integration of different ontologies is also an important issue. Integration can, for example, be supported as follows: after the knowledge engineer has asserted some inter-ontology relationships, the integrated concept hierarchy is computed and the concepts are checked for consistency. Inconsistent concepts as well as unintended or missing subsumption relationships are thus signs of incorrect or incomplete inter-ontology assertions, which can then be corrected or completed by the knowledge engineer.

Finally, reasoning may also be used when the ontology is deployed. As well as using the pre-computed concept hierarchy, one could, for example, use the ontology to determine the consistency of facts stated in annotations, or infer relationships between annotation instances and ontology classes. More precisely, when searching web pages annotated with terms from the ontology, it may be useful to consider not only exact matches, but also matches with respect to more general or more specific terms – where the latter choice depends on the context. However, in the deployment phase, the requirements on the efficiency of reasoning are much more stringent than in the design and integration phases.

Before arguing why description logics are good candidates for such an ontology language, we provide a brief introduction to and history of description logics.

1.2 Description Logics

Description logics (DLs) [6, 16, 30] are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. 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*, i.e. expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, *logic*-based semantics.

In this introduction, we only illustrate some typical constructors by an example. Formal definitions are given in Sect. 2. Assume that we want to define the concept of "A man that is married to a doctor and has at least five children, all of whom are professors." This concept can be described with the following concept description:

$$\mathsf{Human} \sqcap \neg\mathsf{Female} \sqcap \exists\mathsf{married}.\mathsf{Doctor} \sqcap (\geq 5\,\mathsf{hasChild}) \sqcap \forall\mathsf{hasChild}.\mathsf{Professor}$$

This description employs the Boolean constructors *conjunction* ($\sqcap$), which is interpreted as set intersection, and *negation* ($\neg$), which is interpreted as set complement, as well as the *existential restriction* constructor ($\exists R.C$), the *value restriction* constructor ($\forall R.C$), and the *number restriction* constructor ($\geq n\,R$). An individual, say Bob, belongs to $\exists\mathsf{married}.\mathsf{Doctor}$ if there exists an individual that is married to Bob (i.e. is related to Bob via the $\mathsf{married}$ role) and is a doctor (i.e. belongs to the concept Doctor). Similarly, Bob belongs to $(\geq 5\,\mathsf{hasChild})$ iff he has at least five children, and he belongs to $\forall\mathsf{hasChild}.\mathsf{Professor}$ iff all his children (i.e. all individuals related to Bob via the $\mathsf{hasChild}$ role) are professors.

In addition to this description formalism, DLs are usually equipped with a terminological and an assertional formalism. In its simplest form, *terminological axioms* can be used to introduce names (abbreviations) for complex descriptions. For example, we could introduce the abbreviation $\mathsf{HappyMan}$ for the concept description from above. More expressive terminological formalisms allow the statement of constraints such as

$$\exists\mathsf{hasChild}.\mathsf{Human} \sqsubseteq \mathsf{Human},$$

which says that only humans can have human children. A set of terminological axioms is called a TBox. The *assertional formalism* can be used to state properties of individuals. For example, the assertions

$$\mathsf{HappyMan}(\mathsf{BOB}), \quad \mathsf{hasChild}(\mathsf{BOB}, \mathsf{MARY})$$

state that Bob belongs to the concept $\mathsf{HappyMan}$ and that Mary is one of his children. A set of such assertions is called an ABox, and the named individuals that occur in ABox assertions are called ABox individuals.

Description logic systems provide their users with various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. The *subsumption* algorithm determines subconcept–superconcept relationships: C is subsumed by D iff all instances of C are necessarily instances of D, i.e. the first description is always interpreted as a subset of the second description. For example, given the definition of HappyMan from above, HappyMan is subsumed by $\exists$hasChild.Professor – since instances of HappyMan have at least five children, all of whom are professors, they also have a child that is a professor. The *instance* algorithm determines instance relationships: the individual i is an instance of the concept description C iff i is always interpreted as an element of C. For example, given the assertions from above and the definition of HappyMan, MARY is an instance of Professor. The *consistency* algorithm determines whether a knowledge base (consisting of a set of assertions and a set of terminological axioms) is non-contradictory. For example, if we add $\neg$Professor(MARY) to the two assertions from above, then the knowledge base containing these assertions together with the definition of HappyMan from above is inconsistent.

In order to ensure reasonable and predictable behavior of a DL system, these inference problems 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. Roughly, the research related to this issue can be classified into the following four phases.

Phase 1 (1980–1990) was mainly concerned with implementation of systems, such as KLONE, K-REP, BACK, and LOOM [24, 70, 71, 80]. These systems employed so-called *structural subsumption algorithms*, which first normalize the concept descriptions, and then recursively compare the syntactic structure of the normalized descriptions [73]. These algorithms are usually relatively efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e. for more expressive DLs they cannot detect all the existing subsumption/instance relationships. At the end of this phase, early formal investigations into the complexity of reasoning in DLs showed that most DLs do not have polynomial-time inference problems [23, 74]. As a reaction, the implementors of the CLASSIC system (the first industrial-strength DL system) carefully restricted the expressive power of their DL [22, 79].

Phase 2 (1990–1995) started with the introduction of a new algorithmic paradigm into DLs, so-called *tableau-based algorithms* [40, 54, 88]. They work on propositionally closed DLs (i.e. DLs with full Boolean operators) and are complete also for expressive DLs. To decide the consistency of a knowledge base, a tableau-based algorithm tries to construct a model of it by breaking

down the concepts in the knowledge base, thus inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed with obvious contradictions, or it stops with a "canonical" model. Since in propositionally closed DLs, subsumption and satisfiability can be reduced to consistency, a consistency algorithm can solve all the inference problems mentioned above. The first systems employing such algorithms (KRIS and CRACK) demonstrated that optimized implementations of these algorithm lead to an acceptable behavior of the system, even though the worst-case complexity of the corresponding reasoning problems is no longer in polynomial time [9, 27]. This phase also saw a thorough analysis of the complexity of reasoning in various DLs [39–41]. Another important observation was that DLs are very closely related to modal logics [85].

Phase 3 (1995–2000) is characterized by the development of inference procedures for very expressive DLs, either based on the tableau-approach [58, 60] or on a translation into modal logics [35–38]. Highly optimized systems (FaCT, RACE, and DLP [50, 55, 78]) showed that tableau-based algorithms for expressive DLs lead to a good practical behavior of the system even on (some) large knowledge bases. In this phase, the relationship to modal logics [36, 86] and to decidable fragments of first-order logic was also studied in more detail [19, 45–47, 76], and applications in databases (like schema reasoning, query optimization, and integration of databases) were investigated [28, 29, 31].

We are now at the beginning of *Phase 4*, where industrial strength DL systems employing very expressive DLs and tableau-based algorithms are being developed, with applications like the Semantic Web or knowledge representation and integration in bio-informatics in mind.

1.3 Description Logics as Ontology Languages

As already mentioned above, high quality ontologies are crucial for many applications, and their construction, integration, and evolution greatly depends on the availability of a well-defined semantics and powerful reasoning tools. Since DLs provide for both, they should be ideal candidates for ontology languages. That much was already clear ten years ago, but at that time there was a fundamental mismatch between the expressive power and the efficiency of reasoning that DL systems provided, and the expressivity and the large knowledge bases that ontologists needed [42]. Through the basic research in DLs of the last 10–15 years that we have summarized above, this gap between the needs of ontologist and the systems that DL researchers provide has finally become narrow enough to build stable bridges.

The suitability of DLs as ontology languages has been highlighted by their role as the foundation for several web ontology languages, including OWL, an ontology language standard developed by the W3C Web-Ontology Working Group[2] (see chapter "Web Ontology Language: OWL"). OWL has a syntax

[2] `http://www.w3.org/2001/sw/WebOnt/`

based on RDF Schema, but the basis for its design is the expressive DL $\mathcal{SHIQ}$ [60],[3] and the developers have tried to find a good compromise between expressiveness and the complexity of reasoning. Although reasoning in $\mathcal{SHIQ}$ is decidable, it has a rather high worst-case complexity (ExpTime). Nevertheless, highly optimized $\mathcal{SHIQ}$ reasoners such as FaCT++ [95], Racer [52] and Pellet [91] behave quite well in practice.

Let us point out some of the features of $\mathcal{SHIQ}$ that make this DL expressive enough to be used as an ontology language. Firstly, $\mathcal{SHIQ}$ provides number restrictions that are more expressive than the ones introduced above (and employed by earlier DL systems). With the *qualified number restrictions* available in $\mathcal{SHIQ}$, as well as being able to say that a person has at most two children (without mentioning the properties of these children):

$$(\leq 2\,\mathsf{hasChild}),$$

one can also specify that there is at most one son and at most one daughter:

$$(\leq 1\,\mathsf{hasChild}.\neg\mathsf{Female}) \sqcap (\leq 1\,\mathsf{hasChild}.\mathsf{Female}).$$

Secondly, $\mathcal{SHIQ}$ allows the formulation of complex terminological axioms like "humans have human parents":

$$\mathsf{Human} \sqsubseteq \exists \mathsf{hasParent}.\mathsf{Human}.$$

Thirdly, $\mathcal{SHIQ}$ also allows for *inverse roles*, *transitive roles*, and *subroles*. For example, in addition to hasChild one can also use its inverse hasParent, one can specify that hasAncestor is transitive, and that hasParent is a subrole of hasAncestor.

It has been argued in the DL and the ontology community that these features play a central role when describing properties of aggregated objects and when building ontologies [43, 83, 93]. The actual use of a DL providing these features as the underlying logical formalism of the web ontology language OWL [57] substantiates this claim [93].[4]

Finally, we would like to briefly mention three extensions to $\mathcal{SHIQ}$ that are often used in ontology languages (we will discuss them in more detail in Sect. 4).

Complex roles are often required in ontologies. For example, when describing complex physically composed structures it may be desirable to express the fact that damage to a part of the structure implies damage to the structure as a whole. This feature is particularly important in medical ontologies: it is supported in the Grail DL [81], which was specifically designed for use with medical terminology, and in another medical terminology application using

[3] To be exact, it is based on $\mathcal{SHOIN}$.

[4] The more expressive qualified number restrictions are not supported by OWL, but are featured in the proposed OWL 2 extension (see Sect. 4).

the comparatively inexpressive DL $\mathcal{ALC}$, a rather complex "work around" is performed in order to capture this kind of information [89].[5]

It is quite straightforward to extend $\mathcal{SHIQ}$ so that this kind of propagation can be expressed: simply allow for the use of complex roles in role inclusion axioms. E.g. $\mathsf{hasLocation} \circ \mathsf{partOf} \sqsubseteq \mathsf{hasLocation}$ expresses the fact that things located in part of something are also located in the thing as a whole. Although this leads to undecidability in general, syntactic restrictions can be devised that lead to a decidable logic [59].

Concrete domains [7,69] integrate DLs with concrete sets such as the real numbers, integers, or strings, and built-in predicates such as comparisons $\leq$, comparisons with constants ≤ 17, or $\mathsf{isPrefixOf}$. This supports the modelling of concrete properties of abstract objects such as the age, the weight, or the name of a person, and the comparison of these concrete properties. Unfortunately, in their unrestricted form, concrete domains can have dramatic effects on the decidability and computational complexity of the underlying DL [69].

Nominals are special concept names that are to be interpreted as singleton sets. Using a nominal Turing, we can describe all those computer scientists that have met Turing by $\mathsf{CSientist} \sqcap \exists \mathsf{hasMet}.\mathsf{Turing}$. Again, nominals can have dramatic effects on the complexity of a logic [94]. The extension of $\mathcal{SHIQ}$ with nominals is usually called $\mathcal{SHOIQ}$.

2 The Expressive Description Logic $\mathcal{SHIQ}$

In this section, we present syntax and semantics of the expressive DL $\mathcal{SHIQ}$ [60] (although, as can be seen in chapter "Web Ontology Language: OWL", the DL underlying OWL is, in some respects, slightly more expressive).

In contrast to most of the DLs considered in the literature, which concentrate on constructors for defining concepts, the DL $\mathcal{SHIQ}$ also allows for rather expressive roles. Of course, these roles can then be used in the definition of concepts.

Definition 1 (Syntax and semantics of $\mathcal{SHIQ}$-roles and concepts). *Let* $\mathbf{R}$ *be a set of* role names*, which is partitioned into a set* $\mathbf{R}_+$ *of transitive roles and a set* $\mathbf{R}_\mathsf{P}$ *of normal roles. The set of all* $\mathcal{SHIQ}$-roles *is* $\mathbf{R} \cup \{r^- \mid r \in \mathbf{R}\}$*, where* r^- *is called the* inverse *of the role* r*.*

Let $\mathbf{C}$ *be a set of* concept names*. The set of* $\mathcal{SHIQ}$-concepts *is the smallest set such that*

1. *every concept name* $A \in \mathbf{C}$ *is a* $\mathcal{SHIQ}$*-concept,*
2. *if* C *and* D *are* $\mathcal{SHIQ}$*-concepts and* r *is a* $\mathcal{SHIQ}$*-role, then* $C \sqcap D$*,* $C \sqcup D$*,* $\neg C$*,* $\forall r.C$*, and* $\exists r.C$ *are* $\mathcal{SHIQ}$*-concepts,*

[5] In this approach, so-called *SEP-triplets* are used both to compensate for the absence of transitive roles in $\mathcal{ALC}$, and to express the propagation of properties across a distinguished "part-of" role.

3. if C is a $\mathcal{SHIQ}$-concept, r is a simple[6] *$\mathcal{SHIQ}$-role, and $n \in \mathbb{N}$, then $(\leqslant n\, r.C)$ and $(\geqslant n\, r.C)$ are $\mathcal{SHIQ}$-concepts.*

An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a set $\Delta^{\mathcal{I}}$, called the* domain *of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ that maps every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ such that, for all $p \in \mathbf{R}$ and $r \in \mathbf{R}_+$,*

$$\langle x, y\rangle \in p^{\mathcal{I}} \quad \text{iff} \quad \langle y, x\rangle \in (p^-)^{\mathcal{I}},$$
$$\text{if } \langle x, y\rangle \in r^{\mathcal{I}} \text{ and } \langle y, z\rangle \in r^{\mathcal{I}} \text{ then } \langle x, z\rangle \in r^{\mathcal{I}}.$$

The interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ maps, additionally, every concept to a subset of $\Delta^{\mathcal{I}}$ such that

$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}, \qquad (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}, \qquad \neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}},$$
$$(\exists r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{There is some } y \in \Delta^{\mathcal{I}} \text{ with } \langle x, y\rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\},$$
$$(\forall r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \text{For all } y \in \Delta^{\mathcal{I}}, \text{ if } \langle x, y\rangle \in r^{\mathcal{I}}, \text{then } y \in C^{\mathcal{I}}\},$$
$$(\leqslant n\, r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp r^{\mathcal{I}}(x, C) \leqslant n\},$$
$$(\geqslant n\, r.C)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \sharp r^{\mathcal{I}}(x, C) \geqslant n\},$$

where $\sharp M$ denotes the cardinality of the set M, and $r^{\mathcal{I}}(x, C) := \{y \mid \langle x, y\rangle \in r^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\}$. If $x \in C^{\mathcal{I}}$, then we say that x is an instance *of C in $\mathcal{I}$, and if $\langle x, y\rangle \in r^{\mathcal{I}}$, then y is called an* r-successor *of x in $\mathcal{I}$.*

So far, we have fixed the syntax and semantics of concepts and roles. Next, we define how they can be used in a $\mathcal{SHIQ}$ TBox. Please note that authors sometimes distinguish between a role hierarchy or RBox and a TBox – we do not make this distinction here.

Definition 2 (TBox). *A* role inclusion axiom *is of the form $r \sqsubseteq s$, where r, s are $\mathcal{SHIQ}$-roles. A* general concept inclusion *(GCI) is of the form $C \sqsubseteq D$, where C, D are $\mathcal{SHIQ}$-concepts.*

A finite set of role inclusion axioms and GCIs is called a TBox.

An interpretation $\mathcal{I}$ is a model *of a TBox $\mathcal{T}$ if it satisfies all axioms in $\mathcal{T}$, i.e. $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds for each $C \sqsubseteq D \in \mathcal{T}$ and $r^{\mathcal{I}} \subseteq s^{\mathcal{I}}$ holds for each $r \sqsubseteq s \in \mathcal{T}$.*

A concept definition is of the form $A \equiv C$, where A is a concept name; it can be seen as an abbreviation for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$.

In addition to describing the relevant notions of an application domain, a DL knowledge base may also contain knowledge about the properties of specific individuals (or objects) existing in this domain. This done in the assertional part of the knowledge base (ABox).

[6] We refer the interested reader to [60] for a definition of simple roles: roughly speaking, a role is simple if it is neither transitive nor has a transitive sub-role. Only simple roles are allowed in number restrictions to ensure decidability [60].

Definition 3. *Let* $\mathbf{I}$ *be a set of* individual names *disjoint from* $\mathbf{R}$ *and* $\mathbf{C}$*. For* $a, b \in \mathbf{I}$ *individual names,* C *a possibly complex* $\mathcal{SHIQ}$ *concept, and* r *a* $\mathcal{SHIQ}$ *role, an expression of the form*

- $C(a)$ *is called a* concept assertion*, and*
- $r(a, b)$ *is called a* role assertion*.*

A finite set of concept and role assertions is called an ABox*.*

An interpretation function $\cdot^{\mathcal{I}}$*, additionally, is required to map every individual name* $a \in \mathbf{I}$ *to an element* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*. An interpretation* $\mathcal{I}$ satisfies

- *a concept assertion* $C(a)$ *if* $a^{\mathcal{I}} \in C^{\mathcal{I}}$*, and*
- *a role assertion* $r(a, b)$ *if* $\langle a^{\mathcal{I}}, b^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$*.*

An interpretation that satisfies each concept assertion and each role assertion in an ABox $\mathcal{A}$ *is called a* model *of* $\mathcal{A}$*.*

Inference problems for concepts are defined w.r.t. a TBox. Inference problems for individuals additionally involve an ABox.

Definition 4. *The concept* C *is called* satisfiable with respect to the TBox $\mathcal{T}$ *iff there is a model* $\mathcal{I}$ *of* $\mathcal{T}$ *with* $C^{\mathcal{I}} \neq \emptyset$*. Such an interpretation is called a* model of C *w.r.t.* $\mathcal{T}$*. The concept* D subsumes *the concept* C *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* $\mathcal{T}$*. Two concepts* C, D *are* equivalent *w.r.t.* $\mathcal{T}$ *(written* $C \equiv_{\mathcal{T}} D$*) if they subsume each other.*

The ABox $\mathcal{A}$ *is called* consistent with respect to the TBox $\mathcal{T}$ *iff there exists a model of* $\mathcal{T}$ *and* $\mathcal{A}$*. The individual* a *is called an* instance of the concept C with respect to the TBox $\mathcal{T}$ and the ABox $\mathcal{A}$ *iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$ *holds for all models of* $\mathcal{I}$ *of* $\mathcal{T}$ *and* $\mathcal{A}$*.*

By definition, equivalence can be reduced to subsumption. In addition, subsumption can be reduced to satisfiability since $C \sqsubseteq_{\mathcal{T}} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}$. Satisfiability and the instance problem can be reduced to the consistency problem since C is satisfiable w.r.t. $\mathcal{T}$ if the ABox $\{C(a)\}$ is consistent w.r.t. $\mathcal{T}$, and a is an instance of C w.r.t. $\mathcal{T}$ and $\mathcal{A}$ if the ABox $\mathcal{A} \cup \{\neg C(a)\}$ is inconsistent w.r.t. $\mathcal{T}$.

As mentioned above, most DLs are (decidable) fragments of (first-order) predicate logic [5, 19]. Viewing role names as binary relations, concept names as unary relations, and individual names as constants, for example, the role inclusion axiom $r \sqsubseteq s^-$ translates into $\forall x \forall y.r(x, y) \Rightarrow s(y, x)$, the concept assertion $(A \sqcap B)(a)$ translates into $A(a) \wedge B(a)$, and the GCI $A \sqcap \exists r.C \sqsubseteq D \sqcup \forall s^-.E$ translates into

$$\forall x.(A(x) \wedge \exists y.r(x, y) \wedge C(y)) \Rightarrow (D(x) \vee \forall y.s(y, x) \Rightarrow E(y)).$$

This translation preserves the semantics: we can easily view DL interpretations as predicate logic interpretations, and then prove, e.g. that each model of a concept C w.r.t. a TBox $\mathcal{T}$ is a model of the translation of C conjoined with the (universally quantified) translations of $\mathcal{T}$.

The reasoning services that can decide the inference problems introduced above can be implemented using various algorithmic techniques, including tableaux-based techniques (see chapter "Tableau-Based Reasoning") and resolution-based techniques (see "Resolution-Based Reasoning for Ontologies").

3 Describing Ontologies in $\mathcal{SHIQ}$

In general, an ontology can be formalised in a DL knowledge base as follows. Firstly, we restrict the possible worlds by introducing restrictions on the allowed interpretations. For example, to express that, in our world, we want to consider humans, which are either muggles or sorcerers, we can use the GCIs

$$\mathsf{Human} \sqsubseteq \mathsf{Muggle} \sqcup \mathsf{Sorcerer} \quad \text{and} \quad \mathsf{Muggle} \sqsubseteq \neg\mathsf{Sorcerer}.$$

Next, to express that humans have exactly two parents and that all parents and children of humans are human, we can use the following GCI:

$$\mathsf{Human} \sqsubseteq \forall \mathsf{hasParent}.\mathsf{Human} \sqcap (\leqslant 2\ \mathsf{hasParent}.\top) \sqcap (\geqslant 2\ \mathsf{hasParent}.\top) \sqcap \forall \mathsf{hasParent}^{-}.\mathsf{Human},$$

where $\top$ is an abbreviation for the top concept $A \sqcup \neg A$.[7]

In addition, we consider the *transitive* role $\mathsf{hasAncestor}$, and the role inclusion

$$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}.$$

The next GCI expresses that humans having an ancestor that is a sorcerer are themselves sorcerers:

$$\mathsf{Human} \sqcap \exists \mathsf{hasAncestor}.\mathsf{Sorcerer} \sqsubseteq \mathsf{Sorcerer}.$$

Secondly, we can define the relevant notions of our application domain using concept definitions. Recall that the concept definition $A \equiv C$ stands for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$. A concept name is called *defined* if it occurs on the left-hand side of a definition, and *primitive* otherwise.

We want our concept definitions to have definitional impact, i.e. the interpretation of the primitive concept and role names should uniquely determine the interpretation of the defined concept names. For this, the set of concept definitions together with the additional GCIs must satisfy three conditions:

1. There are no multiple definitions, i.e. each defined concept name must occur at most once as the left-hand side of a concept definition.

[7] When the qualifying concept is $\top$, this is equivalent to an unqualified restriction, and it will often be written as such, e.g. $(\leqslant 2\ \mathsf{hasParent})$.

2. There are no cyclic definitions, i.e. no cyclic dependencies between the defined names in the set of concept definitions.[8]
3. The defined names do not occur in any of the additional GCIs.

In contrast to concept definitions, the GCIs in $\mathcal{SHIQ}$ may well have cyclic dependencies between concept names. An example are the above GCIs describing humans.

As a simple example of a set of concept definitions satisfying the restrictions from above, we define the concepts grandparent and parent[9]:

$$\mathsf{Parent} \equiv \mathsf{Human} \sqcap \exists \mathsf{hasParent}^{-}.\top,$$
$$\mathsf{Grandparent} \equiv \exists \mathsf{hasParent}^{-}.\mathsf{Parent}.$$

The TBox consisting of the above concept definitions and GCIs, together with the fact that hasAncestor is a transitive superrole of hasParent, implies the following subsumption relationship:

$$\mathsf{Grandparent} \sqcap \mathsf{Sorcerer} \sqsubseteq \exists \mathsf{hasParent}^{-}.\exists \mathsf{hasParent}^{-}.\mathsf{Sorcerer},$$

i.e. grandparents who are sorcerers have a grandchild who is a sorcerer. Though this conclusion may sound reasonable given the assumptions, it requires quite some reasoning to obtain it. In particular, one must use the fact that hasAncestor (and thus also $\mathsf{hasAncestor}^{-}$) is transitive, that $\mathsf{hasParent}^{-}$ is the inverse of hasParent, and that we have a GCI that says that children of humans are again humans.

To sum up, a $\mathcal{SHIQ}$-TBox can, on the one hand, axiomatize the basic notions of an application domain (the primitive concepts) by GCIs, transitivity statements, and role inclusions, in the sense that these statements restrict the possible interpretations of the basic notions. On the other hand, more complex notions (the defined concepts) can be introduced by concept definitions. Given an interpretation of the basic notions, the concept definitions uniquely determine the interpretation of the defined notions.

The *taxonomy* of such a TBox is then given by the subsumption hierarchy of the defined concepts. It can be computed using a subsumption algorithm for $\mathcal{SHIQ}$ (see chapters "Tableau-Based Reasoning" and "Resolution-Based Reasoning for Ontologies"). The knowledge engineer can test whether the TBox captures her intuition by checking the satisfiability of the defined concepts (since it does not make sense to give a complex definition for the empty concept), and by checking whether their place in the taxonomy corresponds to their intuitive place. The taxonomy of our example TBox would contain, for example, the fact that Grandparent is subsumed by Parent which is, in turn, subsumed by Human – if this is not intended, then the knowledge engineer

[8] In order to give cyclic definitions definitional impact, one would need to use fixpoint semantics for them [1, 75].

[9] In addition to the role hasParent, which relates children to their parents, we use the concept Parent, which describes all humans having children.

would need to go back and modify the TBox. The expressive power of $\mathcal{SHIQ}$ together with the fact that one can "verify" the TBox in the sense mentioned above is the main reason for $\mathcal{SHIQ}$ being well-suited as an ontology language [43, 83, 93].

In case we have, in addition to our TBox $\mathcal{T}$, also an ABox $\mathcal{A}$, we can first ask a DL reasoner to check whether $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$ to make sure that our assertions in $\mathcal{A}$ conform with the axioms expressed in $\mathcal{T}$. Consider the following ABox:

$$\mathcal{A} = \{\, \mathsf{Human}(\mathsf{Harry}),\ \mathsf{Sorcerer}(\mathsf{Bob}) \\ \mathsf{hasParent}(\mathsf{Harry}, \mathsf{Bob})\},$$

and let $\mathcal{T}$ consist of all axioms in this section. We can first use a DL reasoner to prove that $\mathcal{A}$ is consistent w.r.t. $\mathcal{T}$. Next, we can *query* $\mathcal{A}$ through $\mathcal{T}$. For example, we can ask a DL reasoner to retrieve all instances of Human w.r.t. $\mathcal{T}$ and $\mathcal{A}$. This would result in Harry and Bob being returned: for the former, this information is explicit in $\mathcal{A}$, for the latter, this is implied by the GCI which states that parents of humans are humans. Similarly, both Harry and Bob are instances of Sorcerer w.r.t. $\mathcal{T}$ and $\mathcal{A}$: for Harry, this is a consequence of the GCI which states that offsprings of sorcerers are sorcerers. As a final example, let us point out that our ABox contains no instance of $\forall$hasParent.Sorcerer: even though all *explicitly known* parents of Harry are sorcerers, Harry could have other parents (and indeed must have another parent) who may or may not be a sorcerer – this feature of DL semantics is known as the "open world assumption" [5].

4 Extensions and Variants of $\mathcal{SHIQ}$

The ontology language OWL extends $\mathcal{SHIQ}$ with nominals and concrete datatypes; see chapter "Web Ontology Language: OWL." In this section, we discuss the consequences of these extensions on the reasoning problems in $\mathcal{SHIQ}$.

Concrete datatypes, as available in OWL, are a very restricted form of concrete domains [7]. For example, using the concrete domain of all nonnegative integers equipped with the $<$ predicate, a (functional) role age relating (abstract) individuals to their (concrete) age, and a (functional) subrole father of hasParent, the following axiom states that children are younger than their fathers:

$$\mathsf{Animal} \sqsubseteq (\mathsf{age} < (\mathsf{father} \circ \mathsf{age})).$$

Extending expressive DLs with concrete domains may easily lead to undecidability [8, 68]. In OWL, however, no datatype predicates are supported – only XML schema datatypes (such as integer and string) and enumerations (such as $\{1, 2, 5, 7\}$) can be used in descriptions. These restrictions are enough to

ensure that decidability is not compromised (in fact in [77], decidability of $\mathcal{SHIQ}$ extended with a more general type of concrete domains is shown).

Concerning nominals, things become a bit more complicated: nominals are individual names used as concepts, as in $\mathsf{Catholic} \sqcap \exists \mathsf{hasSeen}.\{\mathsf{Pope}\}$ and thus allow the use of individuals not only in ABoxes, but also in concept expressions and TBoxes. Firstly, we can use the same (relativised axiomatization) technique as used for $\mathcal{SHIQ}$ in [94] to translate $\mathcal{SHIQ}$ extended with nominals into a fragment of C2, the two-variable fragment of first order logic with counting quantifiers [48, 76]. Since this translation is polynomial, satisfiability and subsumption are decidable in NExpTime. This is optimal since the problem is also NExpTime-hard [94]. Roughly speaking, the combination of GCIs (or transitive roles and role inclusions), inverse roles, and number restrictions with nominals is responsible for this leap in complexity (from ExpTime for $\mathcal{SHIQ}$ to NExpTime). Until recently, no "practical" decision procedure for $\mathcal{SHOIQ}$, i.e. the extension of $\mathcal{SHIQ}$ with nominals, had been described, where by "practical" we mean a decision procedure that works in some "goal-directed" way, in contrast to "blindly" guessing a model $\mathcal{I}$ of at most exponential size and then checking whether $\mathcal{I}$ is indeed a model of the input. An extension of the tableaux algorithm for $\mathcal{SHIQ}$ has, however, now been developed [59], has been successfully implemented in the FaCT++ and Pellet systems, and seems to work well on realistic ontologies [90].

Finally, as mentioned above, it is quite straightforward to extend $\mathcal{SHIQ}$, or even $\mathcal{SHOIQ}$, with complex role inclusion axioms. The resulting DL, $\mathcal{SROIQ}$ [56], is the basis for a recent proposal to extend the OWL language, the extended language being called OWL 2.[10] In addition to complex role inclusion axioms, OWL 2 also supports qualified number restrictions, and more expressive datatypes than OWL.

5 Reasoning Beyond the Standard Inference Problems

As argued in the introduction, standard reasoning services for concepts (such as satisfiability and subsumption algorithms) can be used in different phases of the ontology life cycle. In the design phase, they can test whether concepts are non-contradictory and can derive implied relations between concepts. However, for these services to be applied, one already needs a sufficiently developed TBox. The result of reasoning can then be used to develop the TBox further. Until recently, however, DL systems provided no reasoning support for writing this initial TBox. The development of so-called non-standard inferences in DLs (like computing least common subsumers [13,32,65,67], most specific concepts [10, 66], rewriting [14], approximation [26], and matching [3, 11, 12, 20]) tries to overcome this deficit. These kinds of inferences are sketched in the first subsection.

[10] `http://www.w3.org/TR/2008/WD-owl2-syntax-20081202/`

In the presence of ABoxes, one often wants to ask queries that are more complex than simple instance queries involving only one individual and one concept. So-called conjunctive queries, which are treated in the second subsection, overcome this deficit.

5.1 Non-standard Inferences

In this subsection, we will sketch how non-standard inferences can support building a DL knowledge base.

Assume that the knowledge engineer wants to introduce the definition of a new concept into the TBox. In many cases, she will not develop this new definition from scratch, but rather try to re-use things that are already present in some knowledge base (either the one she is currently building or a previous one). In a chemical process engineering application [72, 82], we have observed two ways in which this is realized in practice:

1. The knowledge engineer decides on the basic structure of the newly defined concept, and then tries to find already defined concepts that have a similar structure. These concepts can then be modified to obtain the new concept.
2. Instead of directly defining the new concept, the knowledge engineer first gives examples of objects that belong to the concept to be defined, and then tries to generalize these examples into a concept definition.

Both approaches can be supported by the non-standard inferences mentioned above, though this kind of support is not yet provided by any of the existing DL systems.

The first approach can be supported by matching concept patterns against concept descriptions. A *concept pattern* is a concept description that may contain variables that stand for descriptions. A *matcher* σ of a pattern D onto the description C replaces the variables by concept descriptions such that the resulting concept $\sigma(D)$ is equivalent to C. For example, assume that the knowledge engineer is looking for concepts concerned with individuals having a son and a daughter sharing some characteristic. This can be expressed by the pattern

$$\exists \mathsf{hasChild}.(\mathsf{Male} \sqcap X) \sqcap \exists \mathsf{hasChild}.(\mathsf{Female} \sqcap X).$$

The substitution $\sigma = \{X \mapsto \mathsf{Tall}\}$ shows that this pattern matches the description $\exists \mathsf{hasChild}.(\mathsf{Male} \sqcap \mathsf{Tall}) \sqcap \exists \mathsf{hasChild}.(\mathsf{Female} \sqcap \mathsf{Tall})$. Note, however, that in some cases the existence of a matcher is not so obvious.

The second approach can be supported by algorithms that compute most specific concepts and least common subsumers. Assume that the examples are given as ABox individuals $i_1, \ldots, i_k$. In a first step, these individuals are generalized into concepts by respectively computing the most specific (w.r.t. subsumption) concepts $C_1, \ldots, C_k$ in the available DL that have these individuals as instances. In a second step, these concepts are generalized into

one concept by computing the least common subsumer of $C_1, \ldots, C_k$, i.e. the least concept description (in the available DL) that subsumes $C_1, \ldots, C_k$. In this context, rewriting of concepts comes into play since the concept descriptions produced by the algorithms for computing least common subsumers may be rather large (and thus not easy to comprehend and modify for the knowledge engineer). Rewriting minimizes the size of these description without changing their meaning by introducing names defined in the TBox.

Until now, the results on such non-standard inferences are restricted to DLs that are considerably less expressive than $\mathcal{SHIQ}$. For some of them, they only make sense if used for inexpressive DLs. For example, in DLs that contain the disjunction constructor, the least common subsumer of $C_1, \ldots, C_k$ is simply their disjunction, and computing this is of no help to the knowledge engineer. What one would like to obtain as a result of the least common subsumer computation are the structural similarities between the input concepts.

Thus, support by non-standard inferences can only be given if one uses DLs of restricted expressive power. However, this also makes sense in the context of ontology engineering. In fact, the users that will require the most support are the naive ones, and it is reasonable to assume that they will not use (or even be offered) the full expressive power of the underlying DL. This two-level approach is already present in tools like Protégé [64], which offer a frame-like user interface. Using this simple interface, one gets only a fragment of the expressive power of OWL. To use the full expressive power, one must type in DL expressions.

Another way to overcome the gap between DLs of different expressive power is to use the approximation inference [26]. Here, one tries to approximate a given concept description C in an expressive DL $\mathcal{L}_1$ by a description D in a less expressive DL $\mathcal{L}_2$. When approximating from above, D should be the least description in $\mathcal{L}_2$ subsuming C, and when approximating from below, D should be the greatest description $\mathcal{L}_2$ subsumed by C.

5.2 Queries

As we have seen in Sect. 3, given an ontology consisting of an ABox and possibly a TBox, we can retrieve instances of concepts from it, thereby explicating knowledge about concept instances in the given ontology. In this sense, we can use concepts as a query language. It has turned out, however, that this is a rather weak query language which does not allow one to query, for example, for humans whose parents are married. Continuing the example from Sect. 3, could express this query as a *conjunctive query*:

$$q(x) :- \mathsf{Human}(x), \mathsf{hasParent}(x, y), \mathsf{hasParent}(x, z), \mathsf{married}(y, z)$$

Conjunctive queries are well-known in the database community and have been suggested as an expressive query language for DLs [29]. Their answers can be sets of individual names from the ABox as in the example query above or,

more generally, sets of tuples (if we have more than one answer variable). Roughly speaking, individual names from the ABox are in the answer set if, for each model of the ontology, we can find a match from the variables into the model's domain such that all conjuncts in the query are satisfied. Hence, as in instance retrieval, all axioms in the ontology are taken fully into account when answering queries. However, in contrast to the standard reasoning problems, and especially instance retrieval, answering conjunctive queries cannot be reduced to consistency. This problem is, however, decidable for a variety of logics [29] and it turned out to remain decidable even if transitive roles are used in the query [44].

6 Conclusion

The emphasis in DL research on a well-defined, logic-based semantics and a thorough investigation of the basic reasoning problems, together with the availability of highly optimized systems for very expressive DLs, makes this family of knowledge representation formalisms an ideal starting point for defining ontology languages. The standard reasoning services such as consistency checking, computation of the taxonomy, testing for unsatisfiable concepts, and instance retrieval, are provided by highly optimised, state-of-the-art DL systems for very expressive DLs. Optimizations of these systems for large ABoxes and the implementation of conjunctive query answering algorithms are active research areas.

To be used in practice, the domain expert also needs tools that further support knowledge acquisition (i.e. building ontologies), maintenance (i.e. evolution of ontologies), and integration and inter-operation of ontologies. First steps in this direction have already been taken. For example, Protégé [64] and SWOOP [63] are tools that support the development of OWL ontologies. On a more fundamental level, non-standard inferences that support building and maintaining knowledge bases are now important topics of DL research. These include the inference problems discussed in Sect. 5.1 but also others that we have not discussed there due to space limitations: for example, tool support has been developed to explain subsumption and unsatisfiability and to repair unsatisfiable concepts (for example, see [62,87]) and to support modular design and re-use of ontologies (for example, see [33]).

In this chapter we have concentrated on very expressive Description Logics that are the formal basis for the web ontology language OWL. For the sake of completeness, we mention here some recent results on inexpressive DLs that are relevant in the context of ontology applications. Several biomedical ontologies, such as SNOMED [92] and the Gene Ontology [34], are based on rather inexpressive DLs, whose main distinguishing feature is that they disallow value restrictions ($\forall r.C$), but provide for existential restrictions ($\exists r.C$). Recently, it has turned out that such inexpressive DLs with existential restrictions behave much better w.r.t. computational complexity than

the corresponding DLs with value restrictions. For example, the subsumption problem in $\mathcal{EL}$, which allows for conjunction, existential restrictions, and the top concept, stays polynomial in the presence of (cyclic or acyclic) concept definitions [2] and even arbitrary GCIs [25]. In [4] it is shown that these polynomiality results also hold for extensions of $\mathcal{EL}$ by constructors that are of interest for ontology applications, such as the bottom concept (which allows disjointness statements to be formulated), nominals, a restricted form of concrete domains, and a restricted form of so-called role-value maps. A first implementation of the polynomial-time subsumption algorithm for such an extension of $\mathcal{EL}$ behaves well on very large bio-medical ontologies [15].

References

1. Franz Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4): 175–219, 1996.
2. Franz Baader. Terminological cycles in a description logic with existential restrictions. In Georg Gottlob and Toby Walsh, editors, *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI 2003)*, pages 325–330. Morgan Kaufmann, Los Altos, 2003.
3. Franz Baader, Sebastian Brandt, and Ralf Küsters. Matching under side conditions in description logics. In Bernhard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 213–218. Morgan Kaufmann, Seattle, WA, 2001.
4. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.
5. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2003.
6. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2003.
7. Franz Baader and Philipp Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 452–457, 1991.
8. Franz Baader and Philipp Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German Workshop on Artificial Intelligence (GWAI'92)*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143. Springer, Berlin, 1992.
9. Franz Baader and Bernhard Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge (PDK'91)*, volume 567 of *Lecture Notes in Artificial Intelligence*, pages 67–86. Springer, Berlin, 1991.
10. Franz Baader and Ralf Küsters. 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)*, volume 1504 of *Lecture Notes in Computer Science*, pages 129–140. Springer, Berlin, 1998.
11. Franz Baader and Ralf Küsters. Matching in description logics with existential restrictions. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 261–272, 2000.
12. Franz Baader, Ralf Küsters, Alex Borgida, and Deborah L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
13. Franz Baader, Ralf Küsters, and Ralf Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI'99)*, pages 96–101, 1999.
14. Franz Baader, Ralf Küsters, and Ralf Molitor. Rewriting concepts using terminologies. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 297–308, 2000.
15. Franz Baader, Carsten Lutz, and Bontawee Suntisrivaraporn. CEL – a polynomial-time reasoner for life science ontologies. In Ulrich Furbach and Natarajan Shankar, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 287–291. Springer, Berlin, 2006.
16. Franz Baader and Ulrike Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 69(1):5–40, 2001.
17. Sean Bechhofer, Ian Horrocks, Carole Goble, and Robert Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 1–9. CEUR (`http://ceur-ws.org/`), 2001.
18. Tim Berners-Lee. *Weaving the Web*. Harpur, San Francisco, 1999.
19. Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1–2):353–367, 1996.
20. Alexander Borgida and Deborah L. McGuinness. Asking queries about frames. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 340–349, 1996.
21. Pim Borst, Hans Akkermans, and Jan Top. Engineering ontologies. *International Journal of Human-Computer Studies*, 46:365–406, 1997.
22. Ronald J. Brachman. "Reducing" CLASSIC to practice: knowledge representation meets reality. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'92)*, pages 247–258. Morgan Kaufmann, Los Altos, 1992.
23. Ronald J. Brachman and Hector J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI'84)*, pages 34–37, 1984.
24. Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
25. Sebastian Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and – what else? In Ramon López de Mántaras and Lorenza Saitta, editors, *Proc. of the 16th Eur. Conf. on Artificial Intelligence (ECAI 2004)*, pages 298–302, 2004.
26. Sebastian Brandt, Ralf Küsters, and Anni-Yasmin Turhan. Approximation and difference in description logics. In D. Fensel, F. Giunchiglia, D. McGuiness, and

M.-A. Williams, editors, *Proc. of the 8th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2002)*, pages 203–214. Morgan Kaufmann, Los Altos, 2002.
27. Paolo Bresciani, Enrico Franconi, and Sergio Tessaris. Implementing and testing expressive description logics: preliminary report. In *Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.
28. Martin Buchheit, Francesco M. Donini, Werner Nutt, and Andrea Schaerf. A refined architecture for terminological systems: terminology = schema + views. *Artificial Intelligence*, 99(2):209–260, 1998.
29. Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.
30. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Daniele Nardi. Reasoning in expressive description logics. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, chapter 23, pages 1581–1634. Elsevier, Amsterdam, 2001.
31. Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 2–13, 1998.
32. William W. Cohen, Alex Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In William Swartout, editor, *Proc. of the 10th Nat. Conf. on Artificial Intelligence (AAAI'92)*, pages 754–760. AAAI Press/The MIT Press, Austin, TX, 1992.
33. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, and Ulrike Sattler. Modular reuse of ontologies: theory and practice. *Journal of Artificial Intelligence Research*, 31:273–318, 2008.
34. The Gene Ontology Consortium. Gene ontology: tool for the unification of biology. *Nature Genetics*, 25:25–29, 2000.
35. Giuseppe De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.
36. Giuseppe De Giacomo and Maurizio Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI'94)*, pages 205–212, 1994.
37. Giuseppe De Giacomo and Maurizio Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th Eur. Conf. on Artificial Intelligence (ECAI'94)*, pages 411–415, 1994.
38. Giuseppe De Giacomo and Maurizio Lenzerini. TBox and ABox reasoning in expressive description logics. In *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'96)*, pages 316–327, 1996.
39. Francesco M. Donini, Bernhard Hollunder, Maurizio Lenzerini, Alberto Marchetti Spaccamela, Daniele Nardi, and Werner Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence*, 2–3:309–327, 1992.
40. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'91)*, pages 151–162, 1991.

41. Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 458–463, 1991.
42. Jon Doyle and Ramesh S. Patil. Two theses of knowledge representation: language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence*, 48:261–297, 1991.
43. Dieter Fensel, Frank van Harmelen, Michel Klein, Hans Akkermans, Jeen Broekstra, Christiaan Fluit, Jos van der Meer, Hans-Peter Schnurr, Rudi Studer, John Hughes, Uwe Krohn, John Davies, Robert Engels, Bernt Bremdal, Fredrik Ygge, Thorsten Lau, Bernd Novotny, Ulrich Reimer, and Ian Horrocks. On-to-knowledge: ontology-based tools for knowledge management. In *Proceedings of the eBusiness and eWork 2000 (eBeW'00) Conference*, October 2000.
44. Birte Glimm, Ian Horrocks, Carsten Lutz, and Ulrike Sattler. Conjunctive query answering for the description logic $\mathcal{SHIQ}$. *Journal of Artificial Intelligence Research*, 31:157–204, 2008.
45. Erich Grädel. Guarded fragments of first-order logic: a perspective for new description logics? In *Proc. of the 1998 Description Logic Workshop (DL'98)*. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-11/`, 1998.
46. Erich Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
47. Erich Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.
48. Erich Grädel, Martin Otto, and Eric Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, 1997.
49. Thomas Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2):199–220, 1993.
50. Volker Haarslev and Ralf Möller. RACE system description. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 130–132. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.
51. Volker Haarslev and Ralf Möller. Expressive ABox reasoning with number restrictions, role hierarchies, and transitively closed roles. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 273–284, 2000.
52. Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 701–705. Springer, Berlin, 2001.
53. Patrick Hayes. RDF model theory. W3C Working Draft, April 2002. `http://www.w3.org/TR/rdf-mt/`.
54. Bernhard Hollunder, Werner Nutt, and Manfred Schmidt-Schauß. Subsumption algorithms for concept description languages. In *Proc. of the 9th Eur. Conf. on Artificial Intelligence (ECAI'90)*, pages 348–353. Pitman, London, 1990.
55. Ian Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.
56. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 57–67. AAAI Press, New York, 2006.

57. Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics*, 1(1):7–26, 2003.
58. Ian Horrocks and Ulrike Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
59. Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for $\mathcal{SHOIQ}$. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 448–453, 2005.
60. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Journal of the Interest Group in Pure and Applied Logic*, 8(3):239–264, 2000.
61. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In David McAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE 2000)*, volume 1831 of *Lecture Notes in Computer Science*, pages 482–496. Springer, Berlin, 2000.
62. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, and Bernardo Cuenca-Grau. Hendler. Repairing Unsatisfiable Concepts in OWL Ontologies. In *Proc. of 3rd Europ. Semantic Web Conf. (ESWC 2006)*, number 4011 of LNCS, Springer, Berlin, 2006.
63. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau, and James Hendler. SWOOP: a web ontology editing browser. *Journal of Web Semantics*, 4(2), 2005.
64. Holger Knublauch, Ray Fergerson, Natalya Noy, and Mark Musen. The Protégé OWL Plugin: an open development environment for semantic web applications. In Sheila A. McIlraith, Dimitris Plexousakis, and Frank van Harmelen, editors, *Proc. of the 2004 International Semantic Web Conference (ISWC 2004)*, number 3298 in LNCS, pages 229–243. Springer, Berlin, 2004.
65. Ralf Küsters and Alex Borgida. What's in an attribute? Consequences for the least common subsumer. *Journal of Artificial Intelligence Research*, 14:167–203, 2001.
66. Ralf Küsters and Ralf Molitor. Approximating most specific concepts in description logics with existential restrictions. In F. Baader, editor, *Proc. of the Joint German/Austrian Conference on Artificial Intelligence, 24th German / 9th Austrian Conference on Artificial Intelligence (KI 2001)*, volume 2174 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 2001.
67. Ralf Küsters and Ralf Molitor. Computing least common subsumers in ALEN. In Bernard Nebel, editor, *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI 2001)*, pages 219–224. Morgan Kaufmann, Los Altos, 2001.
68. Carsten Lutz. NEXPTIME-complete description logics with concrete domains. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*, pages 45–60. Springer, Berlin, 2001.
69. Carsten Lutz. Description logics with concrete domains – a survey. In *Advances in Modal Logics Volume 4*. World Scientific Publishing Co. Pte. Ltd., Singapore, 2003.
70. Robert MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.

71. Eric Mays, Robert Dionne, and Robert Weida. K-Rep system overview. *SIGART Bulletin*, 2(3):93–97, 1991.
72. Ralf Molitor. *Unterstützung der Modellierung verfahrenstechnischer Prozesse durch Nicht-Standardinferenzen in Beschreibungslogiken*. PhD thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany, 2000. In German.
73. Bernhard Nebel. *Reasoning and Revision in Hybrid Representation Systems*, volume 422 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 1990.
74. Bernhard Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence*, 43:235–249, 1990.
75. Bernhard Nebel. Terminological cycles: semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.
76. Leszek Pacholski, Wieslaw Szwast, and Lidia Tendera. Complexity of two-variable logic with counting. In *Proc. of the 12th IEEE Symp. on Logic in Computer Science (LICS'97)*, pages 318–327. IEEE Computer Society Press, Los Alamitos, CA, 1997.
77. Jeff Z. Pan and Ian Horrocks. Semantic web ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D_n})$ description logic. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, 2002.
78. Peter F. Patel-Schneider. DLP. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, `http://ceur-ws.org/Vol-22/`, 1999.
79. Peter F. Patel-Schneider, Deborah L. McGuiness, Ronald J. Brachman, Lori Alperin Resnick, and Alexander Borgida. The CLASSIC knowledge representation system: guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
80. Christof Peltason. The BACK system – an overview. *SIGART Bulletin*, 2(3):114–119, 1991.
81. A. Rector, S. Bechhofer, C. A. Goble, I. Horrocks, W. A. Nowlan, and W. D. Solomon. The GRAIL concept modelling language for medical terminology. *Artificial Intelligence in Medicine*, 9:139–171, 1997.
82. Ulrike Sattler. *Terminological knowledge representation systems in a process engineering application*. PhD thesis, RWTH Aachen, 1998.
83. Ulrike Sattler. Description logics for the representation of aggregated objects. In *Proc. of the 14th Eur. Conf. on Artificial Intelligence (ECAI 2000)*, 2000.
84. Andrea Schaerf. Reasoning with individuals in concept languages. *Data and Knowledge Engineering*, 13(2):141–176, 1994.
85. Klaus Schild. A correspondence theory for terminological logics: preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI'91)*, pages 466–471, 1991.
86. Klaus Schild. *Querying Knowledge and Data Bases by a Universal Description Logic with Recursion*. PhD thesis, Universität des Saarlandes, Germany, 1995.
87. Stefan Schlobach, Zhisheng Huang, Ronald Cornet, and Frank van Harmelen. Debugging Incoherent Terminologies. *Journal of Automated Reasoning*, 39(3):317–349, 2007.
88. Manfred Schmidt-Schauß and Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
89. Stefan Schulz and Udo Hahn. Parts, locations, and holes – formal reasoning about anatomical structures. In *Proc. of AIME 2001*, volume 2101 of *Lecture Notes in Artificial Intelligence*. Springer, Berlin, 2001.

90. Evren Sirin, Bernardo Cuenca Grau, and Bijan Parsia. From wine to water: optimizing description logic reasoning for nominals. In *Proc. of the 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 90–99. AAAI Press, New York, 2006.
91. Evren Sirin and Bijan Parsia. Pellet: an OWL DL reasoner. In *Proc. of the 2004 Description Logic Workshop (DL 2004)*, 2004.
92. K.A. Spackman, K.E. Campbell, and R.A. Cote. SNOMED RT: a reference terminology for health care. *Journal of the American Medical Informatics Association*, pages 640–644, 1997. Fall Symposium Supplement.
93. Robert Stevens, Ian Horrocks, Carole Goble, and Sean Bechhofer. Building a reasonable bioinformatics ontology using OIL. In *Proceedings of the IJCAI-2001 Workshop on Ontologies and Information Sharing*, pages 81–90. CEUR (`http://ceur-ws.org/`), 2001.
94. Stephan Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001.
95. Dmitry Tsarkov and Ian Horrocks. FaCT++ description logic reasoner: system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, Berlin, 2006.