Reasoning and Query Answering in Description Logics

Magdalena Ortiz and Mantas Šimkus

Institute of Information Systems, Vienna University of Technology
ortiz@kr.tuwien.ac.at, simkus@dbai.tuwien.ac.at

Abstract. Description Logics (DLs) play a central role as formalisms for representing ontologies and reasoning about them. This lecture introduces the basics of DLs. We discuss the knowledge modeling capabilities of some of the most prominent DLs, including expressive ones, and present some DL reasoning services. Particular attention is devoted to the *query answering* problem, and to the increasingly popular framework in which data repositories are queried through DL ontologies. We give an overview of the main challenges that arise in this setting, survey some query answering techniques for both lightweight and expressive DLs, and give an overview of the computational complexity landscape.

Keywords: Description Logics, Query Answering, Ontology Based Data Access.

1 Introduction

Description Logics (DLs) are languages for *Knowledge Representation and Reasoning* [7]. Their function is to allow users to conveniently represent structured knowledge, in such a way that: (1) the representation is non-ambiguous and has a well defined meaning, and (2) *implicit* knowledge can be inferred from the explicitly represented one. DLs are amongst the most popular knowledge representation formalisms, and they have made their way into a whole range of application areas, including health and life sciences [86,48,68], natural language processing [34,35] and data integration [17,72], to name a few. They also play a crucial role in the semantic web, where they provide the foundations of the OWL languages [80,26].

DLs are not one language, but a *family* of languages that differ in their expressivity. To support the inference of implicit knowledge from explicit one, DL reasoners provide *reasoning services*, some of which will be discussed in this tutorial. The efficiency with which reasoners can implement these services is determined by the *computational complexity* of the reasoning problem that is being solved, for the specific DL considered. In general, the complexity of reasoning problems increases with the expressivity of the logic. The many languages in the DL family range from languages that have limited expressivity but where reasoning problems have low complexity, to expressive languages with high complexity

T. Eiter and T. Krennwallner (Eds.): Reasoning Web 2012, LNCS 7487, pp. 1–53, 2012.
© Springer-Verlag Berlin Heidelberg 2012

of reasoning. Research has often focused on understanding this trade-off, and on developing efficient algorithms for reasoning in the different DLs. Much of this tutorial is devoted to studying one specific reasoning service: *query answering*. It is not among the most traditional services but, as we discuss in Section 4, it has gained increasing attention in the last decade and has become a vibrant field of research.

1.1 About This Chapter

The main goal of this chapter is to give an overview of the query answering problem in Description Logics (DLs): its definition, importance, and how it can be solved for different DLs. Since the most prominent lightweight DLs that underlie the OWL 2 profiles are extensively discussed in the tutorial *OWL 2 Profiles: An Introduction to Lightweight Ontology Languages* of this summer school [58], the focus of this chapter is on the so called *expressive DLs*.

The chapter is organized in two main parts:

Introduction to DLs. The first part of the chapter provides a general introduction to DLs, aimed at readers not familiar with them. The introduction is rather basic, yet formal. It does not assume any specific background knowledge, although readers with some elementary background on formal logics may find it easier to follow. Familiarity with the basic notions of classical first order predicate logic (FOL) will be particularly useful, although not strictly necessary.

Survey of Query Answering in DLs. The chapter provides a short survey of the state of the art in query answering: an overview of the problem and its challenges, main techniques, and a brief guide to the literature.

The second part is organized as follows. After introducing the setting of Ontology Based Data Access (OBDA) and illustrating its motivation in Section 4, we give in Section 5 a formal definition of queries and their reasoning problems. In Section 6 we briefly discuss the main challenges that have to be overcome in order to develop query answering algorithms. The state of the art, query answering techniques, and main complexity results obtained until now are presented in Sections 7 and 8. They respectively focus on lightweight and on expressive DLs. Our concluding remarks in Section 9 include a table summarizing the complexity landscape and some pointers to the literature.

Readers familiar with DLs and their traditional reasoning services can skip the first part and proceed directly to Section 4.

Further Reading. The chapter includes a long list of references. For further reading on the topic of the first part of the tutorial, viz. basics of DLs and their reasoning problems, we recommend [7] and its references. For the material on the second part, a more detailed overview of query answering in DLs, with particular

emphasis in expressive ones, as well as more pointers to the literature, can be found in [73].

2 Description Logics, a Big Family of Logics

We give a general introduction to DLs, and present several well-known DLs. We focus on the so-called *expressive DLs* that are obtained by extending the basic DL $\mathcal{ALC}$.

Vocabulary. The very basic building block for describing a domain with DLs is the *vocabulary*, which provides the primitive terms that can be used in the domain's description. Formally, a *DL vocabulary* is a triple $\langle \mathsf{N_R}, \mathsf{N_C}, \mathsf{N_I} \rangle$ of countably infinite, pairwise disjoint sets. The elements of $\mathsf{N_C}$ are called *concept names* and denote the primitive classes that are relevant for the domain of interest. The elements of $\mathsf{N_R}$ are called *role names* and denote the primitive binary relations, and finally, the elements of $\mathsf{N_I}$ are called *individuals* and are used for referring to specific objects in the domain. Throughout the chapter, we assume a fixed DL vocabulary $\langle \mathsf{N_R}, \mathsf{N_C}, \mathsf{N_I} \rangle$. In most of the DLs we consider in this chapter, we can assume that $\mathsf{N_C}$ contains the special concept names $\top$ and $\bot$, respectively called the *top* and *bottom* concepts.

Interpretations. The semantics of DLs is given in terms of *interpretations*, which fix the meaning of the symbols in the vocabulary. An interpretation has a domain which can be any non-empty set, and an interpretation function, which gives meaning to the symbols in the vocabulary. Each individual name is interpreted as one object in the domain. Each concept name, which intuitively corresponds to a set of objects in a class, is naturally interpreted as a subset of the domain. Finally, each role name corresponds to a binary relation between objects. Formally, an *interpretation* for our DL vocabulary is defined as follows.

Definition 1 (Interpretation). *An* interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$*, the* domain *of* $\mathcal{I}$*, and a* valuation function $\cdot^{\mathcal{I}}$ *that maps:*

1. *each individual* $a \in \mathsf{N_I}$ *to an element* $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$*,*
2. *each concept name* $A \in \mathsf{N_C}$ *to a set* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$*,*
3. *each role name* $p \in \mathsf{N_R}$ *to a binary relation* $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$*, and*
4. *for the special concepts names, if present, we have* $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ *and* $\bot^{\mathcal{I}} = \emptyset$*.*

A DL vocabulary is in fact a FOL signature, that contains no function symbols and no variables, but only constants ($\mathsf{N_I}$) and predicates of arities one ($\mathsf{N_C}$) and two ($\mathsf{N_R}$). Interpretations are just standard Tarski-style interpretations as in FOL.

Example 1. In our examples we describe some Greek mythological characters and genealogical relations between them. The common vocabulary that we use contains the following symbols:

- The role names hasParent, hasFather, hasMother, hasAncestor, ...
- The concept names Parent, Mortal, Deity, Male, Female, Hero, ...
- The individual names heracles, zeus, alcmene, perseus, eros, gaia, ...

By convention, concept names start with an upper case and role names with a lower case letter, while individual names are written in lower case Roman font.

Now we introduce the syntax and semantics of the different DLs we consider in this chapter.

2.1 The Basic Description Logics $\mathcal{ALC}$ and $\mathcal{ALCH}$

Each DL offers different combinations of *concept constructors* and *role constructors* that allow us to build complex concept and role expressions from the basic symbols given in the vocabulary.

The DL known as $\mathcal{ALC}$ is considered the 'basic' expressive description logic because it is the minimal one that supports unrestricted use of the basic concept constructors: conjunction, disjunction, negation, and existential and universal restrictions. The first DLs that we introduce in this chapter are $\mathcal{ALC}$ and its extension known as $\mathcal{ALCH}$.

Concepts and Roles. We start with the syntax of *concepts* and *roles*. $\mathcal{ALC}$ and $\mathcal{ALCH}$ do not support any role constructors, that is, only role names p are roles. On the other hand, they provide the five 'basic' concept constructors: negation $\neg C$, conjunction $C_1 \sqcap C_2$, disjunction $C_2 \sqcup C_2$, and existential and universal restrictions which are expressions of the form $\exists p.C$ and $\forall p.C$, respectively.

Definition 2 ($\mathcal{ALCH}$ concepts and roles). *Each role name $p \in \mathsf{N_R}$ is a* role. Concepts C *obey the following grammar, where $A \in \mathsf{N_C}$ and p is a role:*

$$C, C_1, C_2 ::= A \mid \neg C \mid C_1 \sqcap C_2 \mid C_2 \sqcup C_2 \mid \exists p.C \mid \forall p.C$$

In $\mathcal{ALC}$ and all its extensions, the special names $\top$ and $\bot$ can be simulated using a tautological concept of the form $A \sqcup \neg A$ and a contradictory concept of the form $A \sqcup \neg A$, respectively, so it makes no difference whether we assume that they are present in the signature or not.

Concepts of the form $\exists p.\top$ are usually called *unqualified* existential restrictions, and written $\exists p$.

Assertions and Axioms. Using these concepts and role expressions, we can write different kinds of statements. These may also vary in different DLs, but in general, they can be classified into two different kinds:

- At the *extensional level*, we can state that a certain individual participates in some concept, or that some role holds between a pair of individuals; we call this kind of statement *ABox assertions*. A finite set of this assertions is called an *ABox*.

– At the *intensional level*, we can specify general properties of concepts and roles, constraining the way they are interpreted and defining new concepts and roles in terms of other ones. We call these kinds of statements *TBox axioms*, and a *TBox* is a finite set of them. TBoxes are also called *terminologies*.

ABox assertions and TBox axioms together form a *knowledge base* (KB).

Ontologies. The term *ontology* is used frequently, but it does not have a fixed, formally defined meaning. It is used both as a synonym for TBox, or as a synonym for KB. We adopt the former use, i.e., an *ontology* is just a terminology. This meaning is perhaps more frequent, particularly in the context of *ontology based data access* that we will discuss in the second part of this chapter.

We now define the assertions and axioms of the basic DL $\mathcal{ALCH}$.

Definition 3 ($\mathcal{ALCH}$ ABox assertions and TBox axioms). *For $\mathcal{ALCH}$, assertions and axioms are defined as follows.*

ABox assertions*:*

– *If C is a concept and $a \in \mathsf{N_I}$ is an individual, then $C(a)$ is a* concept membership assertion*.*
– *If p is a role and $a, b \in \mathsf{N_I}$ are individuals, then $p(a, b)$ is a* role membership assertion*.*
– *If $a, b \in \mathsf{N_I}$ are individuals, then $a \not\approx b$ is an* inequality assertion*.*

TBox axioms*:*

– *If C_1 and C_2 are concepts, then $C_1 \sqsubseteq C_2$ is a* general concept inclusion axiom (GCI)*.*
– *If p_1 and p_2 are roles, then $p_1 \sqsubseteq p_2$ is a* role inclusion axiom (RIA)*.*

Assertions and axioms for $\mathcal{ALC}$ are defined analogously, but RIAs $p_1 \sqsubseteq p_2$ are disallowed.

Knowledge Bases. Now we can define *knowledge bases*, which are composed by a set of ABox assertions, the ABox, and a set of TBox axioms, the TBox. The definition of these components is the same for all DLs.

Definition 4 (ABoxes, TBoxes, Knowledge bases). *For every DL $\mathcal{L}$, we define:*

– *An* ABox *in $\mathcal{L}$ is a finite set of* ABox assertions *in $\mathcal{L}$.*
– *A* TBox *in $\mathcal{L}$ is a finite set of* TBox axioms *in $\mathcal{L}$.*
– *An* knowledge base (KB) *in $\mathcal{L}$ is a pair $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, where $\mathcal{A}$ is an ABox in $\mathcal{L}$ and $\mathcal{T}$ is a TBox in $\mathcal{L}$.*

Male (zeus)	hasFather (heracles, zeus)
Deity (zeus)	hasMother (heracles, alcmene)
Female (alcmene)	hasFather (alcmene, electryon)
Mortal (alcmene)	hasFather (electryon, perseus)
Hero (heracles)	hasFather (perseus, zeus)

Fig. 1. The Theogony ABox $\mathcal{A}_g$

$$\mathsf{Male} \equiv \neg\mathsf{Female} \tag{1}$$
$$\mathsf{Mortal} \sqsubseteq \neg\mathsf{Deity} \tag{2}$$
$$\mathsf{Primordial} \sqsubseteq \mathsf{Deity} \tag{3}$$
$$\neg\mathsf{Primordial} \sqsubseteq \exists\mathsf{hasFather}.\mathsf{Male} \sqcap \exists\mathsf{hasMother}.\mathsf{Female} \tag{4}$$
$$\mathsf{Deity} \sqsubseteq \forall\mathsf{hasParent}.\mathsf{Deity} \tag{5}$$
$$\mathsf{hasMother} \sqsubseteq \mathsf{hasParent} \tag{6}$$
$$\mathsf{hasFather} \sqsubseteq \mathsf{hasParent} \tag{7}$$

Fig. 2. The $\mathcal{ALCH}$ TBox $\mathcal{T}_1^{theo}$

In many description logics, it is common to define KBs as a triple $\mathcal{K} = \langle \mathcal{A}, \mathcal{T}, \mathcal{R} \rangle$ that additionally contains an RBox, which consists of all axioms that constraint the interpretation of roles. In the case of $\mathcal{ALCH}$, RIAs could be put into the RBox, and in richer logics it may contain more complex expressions. The most prominent DL that contains rich role axioms is $\mathcal{SROIQ}$, the logic that underlies the OWL 2 standard [26]. We will introduce it later on but, to keep the presentation as simple as possible, we will not define RBoxes and instead include role axioms in the TBox for all the considered DLs.

Example 2. Our first knowledge base describing genealogical relations between characters of the Greek mythology is the Theogony knowledge base $\mathcal{K}_1^{theo} = \langle \mathcal{A}^{theo}, \mathcal{T}_1^{theo} \rangle$, which is a knowledge base in $\mathcal{ALCH}$.[1] The ABox $\mathcal{A}^{theo}$ is presented in Figure 1, the TBox $\mathcal{T}_1^{theo}$ can be found in Figure 2. In the examples, we use $E_1 \equiv E_2$ (for E_i a concept or role) as a shortcut for the two axioms $E_1 \sqsubseteq E_2$ and $E_2 \sqsubseteq E_1$.

Intuitively, the assertions in the ABox $\mathcal{A}^{theo}$ indicate that the individual named Zeus is a male deity, while the individual Alcmene is female and mortal. There is an individual named Heracles that is a hero, and who has Zeus as a father and Alcmene as a mother. Alcmene has a father named Electryon, Electryon has Perseus as a father, and Perseus has Zeus as a father.

[1] The examples are only for illustrative purposes and their contents is not necessarily accurate. Some information was taken from `http://www.pantheon.org` and `http://en.wikipedia.org`

The GCI (1) in the TBox $\mathcal{T}_1^{theo}$ ensures that in the models of $\mathcal{K}_1^{theo}$ the domain is partitioned into males and females, and the GCI (2) indicates that mortals cannot be deities. The concept 'primordial' is intended to contain the primordial deities who appeared at the beginning of the universe, and who are ancestors of all other deities; the GCI (3) asserts that they are deities. The GCI (4) indicates that everyone, except the primordial gods, must have a mother and a father. The last GCI (5) says that the parents of a deity must be deities, and the two RIAs make sure that mothers and fathers are also parents.

Example 3. We can consider the $\mathcal{ALC}$ knowledge base that comprises the ABox $\mathcal{A}^{theo}$ and the TBox consisting of axioms (1)–(5), but since the RIAs (6) and (7) in Figure 2 are not allowed in $\mathcal{ALC}$, we cannot relate the roles $\mathsf{hasMother}$ and $\mathsf{hasFather}$ with $\mathsf{hasParent}$. As a result the GCI (5) would not restrict the fillers of the $\mathsf{hasFather}$ and $\mathsf{hasMother}$ relation for a deity to be deities. In this case, the intended meaning of (5) would be captured better by a GCI $\mathsf{Deity} \sqsubseteq \forall \mathsf{hasMother}.\mathsf{Deity} \sqcap \forall \mathsf{hasFather}.\mathsf{Deity}$ instead.

Semantics. In an interpretation, the interpretation function fixes the meaning of the symbols in the vocabulary. This function is extended to all concepts and roles in a given DL $\mathcal{L}$ by means of an inductive definition covering all the concept and role constructors. The definition is such that each concept is mapped to a set of domain elements, and each role to a binary relation over the domain.

Definition 5 (semantics of $\mathcal{ALCH}$ concepts). *Let $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ be an interpretation. The function $\cdot^{\mathcal{I}}$ is inductively extended to all $\mathcal{ALCH}$ concepts as follows:*

$$\begin{aligned}
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C_1 \sqcap C_2)^{\mathcal{I}} &= {C_1}^{\mathcal{I}} \cap {C_2}^{\mathcal{I}} \\
(C_1 \sqcup C_2)^{\mathcal{I}} &= {C_1}^{\mathcal{I}} \cup {C_2}^{\mathcal{I}} \\
(\exists p.C)^{\mathcal{I}} &= \{d \mid \exists d'.(d, d') \in p^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\} \\
(\forall p.C)^{\mathcal{I}} &= \{d \mid \forall d'.(d, d') \in p^{\mathcal{I}} \rightarrow d' \in C^{\mathcal{I}}\}
\end{aligned}$$

Now that we have fixed the semantics of concepts and roles, we can define the satisfaction of assertions and axioms. This is done in a natural way. The symbol $\sqsubseteq$ in the TBox axioms is understood as an 'is-a' relation. That is, a concept inclusion $C_1 \sqsubseteq C_2$ indicates that every object that is C_1 is also C_2, or to be more precise, that every object that participates in the interpretation of concept C_1 also participates in the interpretation of concept C_2. Similarly, a role inclusion $p_1 \sqsubseteq p_2$ indicates that every pair of objects that participates in p_1 also participates in p_2. Concept and role membership assertions in the ABox simply state that (the interpretation of) an individual participates in (the interpretation of) a concept, and that a pair of individuals participates in a role, respectively.

An assertion of the form $a \not\approx b$ states that the individuals a and b cannot be interpreted as the same domain element. This is closely related to the *unique name assumption (UNA)*, sometimes made in related formalisms. Under the UNA, each interpretation $\mathcal{I}$ must be such that $a^{\mathcal{I}} = b^{\mathcal{I}}$ only if $a = b$, that is,

one domain element cannot be the interpretation of two different individuals. In DLs the common practice is not to make the UNA. This setting is more general and, if desired, the UNA can be enforced by adding assertions $a \not\approx b$ for each relevant pair of individuals.

Definition 6 (satisfaction of ABox assertions and TBox axioms). *Let $\mathcal{I}$ be an interpretation. We define the* satisfaction *relation $\mathcal{I} \models \gamma$ for γ an assertion or axiom as follows.*

- *For ABox assertions, we have* $\mathcal{I} \models C(a)$ *if* $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
  $\mathcal{I} \models p(a,b)$ *if* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in p^{\mathcal{I}}$, *and*
  $\mathcal{I} \models a \not\approx b$ *if* $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.
- *For TBox axioms, we have* $\mathcal{I} \models C_1 \sqsubseteq C_2$ *if* $C_1^{\mathcal{I}} \subseteq C_2^{\mathcal{I}}$, *and*
  $\mathcal{I} \models p_1 \sqsubseteq p_2$ *if* $p_1^{\mathcal{I}} \subseteq p_2^{\mathcal{I}}$.

Naturally, the models of an ABox or a TBox are defined as the interpretations that satisfy all the assertions or axioms it contains, and an interpretation is a model of a knowledge base if it is a model of each of its components. This definition is the same for all the DLs that we treat in this chapter.

Definition 7 (satisfaction of ABoxes, TBoxes, and KBs; models). *Let $\mathcal{I}$ be an interpretation. Then*

- *$\mathcal{I}$ satisfies an ABox $\mathcal{A}$, if it satisfies every assertion in $\mathcal{A}$.*
- *$\mathcal{I}$ satisfies a TBox $\mathcal{T}$, if it satisfies every axiom in $\mathcal{T}$.*
- *$\mathcal{I}$ satisfies a knowledge base $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$, if it satisfies $\mathcal{A}$ and $\mathcal{T}$.*

Given an ABox, TBox or knowledge base γ, we write $\mathcal{I} \models \gamma$, if $\mathcal{I}$ satisfies γ, and call $\mathcal{I}$ a model *of γ.*

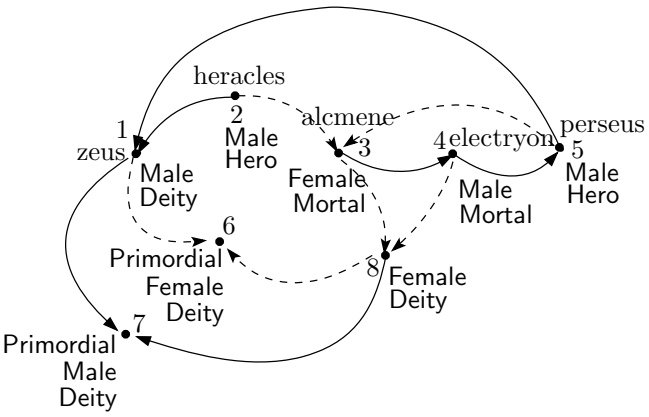

Fig. 3. The interpretation $\mathcal{I}_1$

Example 4. Consider an interpretation $\mathcal{I}_1 = (\Delta^{\mathcal{I}_1}, \cdot^{\mathcal{I}_1})$ with the domain $\Delta^{\mathcal{I}_1} = \{1, \ldots, 8\}$. For the symbols that occur in $\mathcal{K}_1^{theo}$ we have:

$$\begin{array}{rrr} \mathsf{zeus}^{\mathcal{I}_1} = 1 & \mathsf{alcmene}^{\mathcal{I}_1} = 3 & \mathsf{perseus}^{\mathcal{I}_1} = 5 \\ \mathsf{heracles}^{\mathcal{I}_1} = 2 & \mathsf{electryon}^{\mathcal{I}_1} = 4 & \end{array}$$

$$\begin{array}{rrr} \mathsf{Male}^{\mathcal{I}_1} = \{1,2,4,5,7\} & \mathsf{Mortal}^{\mathcal{I}_1} = \{3,4\} & \mathsf{Hero}^{\mathcal{I}_1} = \{2,5\} \\ \mathsf{Female}^{\mathcal{I}_1} = \{3,6,8\} & \mathsf{Deity}^{\mathcal{I}_1} = \{1,6,7,8\} & \mathsf{Primordial}^{\mathcal{I}_1} = \{6,7\} \end{array}$$

$$\begin{array}{rl} \mathsf{hasMother}^{\mathcal{I}_1} & = \{(1,6),(2,3),(3,8),(4,8),(5,3),(8,6)\} \\ \mathsf{hasFather}^{\mathcal{I}_1} & = \{(1,7),(2,1),(3,4),(4,5),(5,1),(8,7)\} \end{array}$$

$$\mathsf{hasParent}^{\mathcal{I}_1} = \mathsf{hasMother}^{\mathcal{I}_1} \cup \mathsf{hasFather}^{\mathcal{I}_1}$$

Note that we can easily represent an interpretation as a node-and-arc-labeled graph, where each node is labeled with the name of the individuals it interprets, and with the set of concepts in whose interpretation it participates.

$\mathcal{I}_1$ is depicted in Figure 3. The hasFather relation is represented by solid arrows, while hasMother is represented by dashed arrows, and hasParent comprises both kinds of arrows. The interpretation $\mathcal{I}_1$ satisfies $\mathcal{A}^{theo}$ and $\mathcal{T}_1^{theo}$, hence $\mathcal{I}_1 \models \mathcal{K}_1^{theo}$.

2.2 Expressive and Lightweight DLs

The term *expressive DLs* usually refers to $\mathcal{ALC}$ and all its extensions. We introduce next two well known families of expressive DLs. As we will discuss later, these logics are very expressive and the computational complexity of reasoning in them is rather high.

In contrast, the term *lightweight DLs* refers to logics that are based on fragments of $\mathcal{ALC}$ and restrict its expressivity to achieve lower complexity, enabling the realization of efficient and scalable algorithms. The most prominent lightweight DLs are the *DL-Lite* and $\mathcal{EL}$ families underlying the OWL QL and EL profiles, respectively. These families are extensively discussed in the tutorial *OWL 2 Profiles: An Introduction to Lightweight Ontology Languages* of this summer school [58], so we will only discuss them briefly in the context of query answering in Section 7.

2.3 The $\mathcal{SH}$ Family

The DL $\mathcal{ALCH}$ can be extended with additional concept and role constructors, and by allowing other kinds of axioms. Some of the most prominent logics obtained this way are the ones in the so called $\mathcal{SH}$ family, which includes, among others, the prominent DLs $\mathcal{SHIQ}$ and $\mathcal{SHOIQ}$.

$\mathcal{SHOIQ}$ is a very expressive DL that is closely related to the Web Ontology Language standard known as OWL-DL [80]. $\mathcal{SHOIQ}$ supports the vast majority of the common DL constructors, and hence most popular DLs can be defined as sublogics of it.

Definition 8 ($\mathcal{SHOIQ}$ concepts and roles). Atomic concepts B, concepts C *and* (atomic) roles P, S *obey the following grammar, where* $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, $p \in \mathsf{N_R}$, *and* $n \geq 0$*:*

$$\begin{aligned} B &::= A \mid \{a\} \\ C, C_1, C_2 &::= B \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \exists P.C \mid \forall P.C \mid \geqslant n\, S.C \mid \leqslant n\, S.C \\ P, S &::= p \mid p^- \end{aligned}$$

The inverse *of* $p \in \mathsf{N_R}$ *is* p^-*, and the inverse of* p^- *is* p*. To avoid expressions such as* $(p^-)^-$*, we denote by* $\mathsf{Inv}(P)$ *the inverse of the role* P*. Concepts of the form* $\{a\}$ *are called* nominals*, while concepts* $\geqslant n\, S.C$ *and* $\leqslant n\, S.C$ *are called* (qualified) number restrictions (NRs)*.*

If a number restriction is of the form $\geqslant n\, S.\top$ or $\leqslant n\, S.\top$, it is called *unqualified* and can be written simply $\geqslant n\, S$ or $\leqslant n\, S$.

In addition to the new role constructor p^- and the new concept constructors $\{a\}$, $\geqslant n\, S.C$ and $\leqslant n\, S.C$, $\mathcal{SHOIQ}$ extends $\mathcal{ALCH}$ with another kind of axioms.

Definition 9 ($\mathcal{SHOIQ}$ ABox assertions and TBox axioms). *ABox assertions, GCIs and RIAs in* $\mathcal{SHOIQ}$ *are defined analogously to* $\mathcal{ALCH}$*, but allowing for* $\mathcal{SHOIQ}$ *concepts and roles where applicable. In addition to GCIs and RIAs,* $\mathcal{SHOIQ}$ *TBoxes allow for* transitivity axioms (TAs)*, which are expressions* $\mathsf{trans}(P)$ *where* P *is a role.*

Knowledge bases in $\mathcal{SHOIQ}$ are defined essentially as for $\mathcal{ALCH}$, but must satisfy an additional constraint: the roles S that occur in the number restrictions $\geqslant n\, S.C$ and $\leqslant n\, S.C$ must be *simple*, which means that they can not be implied by roles occurring in transitivity axioms. Intuitively, this allows us to count only the direct neighbors of a node, but not nodes that are further away in an interpretation. It is well known that dropping this restriction results in an undecidable logic [49].

To formalize the notion of simple roles, we use the relation $\sqsubseteq^{\mathcal{T}}$, which relates each pair of roles P_1, P_2 such that $P_1^{\mathcal{I}} \subseteq P_2^{\mathcal{I}}$ holds in every interpretation that satisfies $\mathcal{T}$.

Definition 10 (simple roles, $\mathcal{SHOIQ}$ knowledge bases). *For a TBox* $\mathcal{T}$*, we denote by* $\sqsubseteq^{\mathcal{T}}$ *the reflexive transitive closure of* $\{(P_1, P_2) \mid P_1 \sqsubseteq P_2$ *or* $\mathsf{Inv}(P_1) \sqsubseteq \mathsf{Inv}(P_2)$ *is in* $\mathcal{T}\}$*; we usually write* $\sqsubseteq^{\mathcal{T}}$ *in infix notation. A role* S *is* simple w.r.t. $\mathcal{T}$*, if there is no* P *such that* $P \sqsubseteq^{\mathcal{T}} S$ *and* $\mathsf{trans}(P) \in \mathcal{T}$*.*

A knowledge base in $\mathcal{SHOIQ}$ *is a pair* $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ *consisting of an ABox* $\mathcal{A}$ *and a TBox* $\mathcal{T}$*, such that all roles* S *occurring in a number restriction* $\leqslant n\, S.C$ *or* $\geqslant n\, S.C$ *are simple w.r.t.* $\mathcal{T}$*.*

Example 5. Consider the $\mathcal{SHOIQ}$ knowledge base $\mathcal{K}_2^{theo} = \langle \mathcal{A}^{theo}, \mathcal{T}_2^{theo} \rangle$, where $\mathcal{A}^{theo}$ is as in Figure 1 and the TBox $\mathcal{T}_2^{theo}$ contains $\mathcal{T}_1^{theo}$ from Figure 2, as well as the GCIs (8)–(10) in Figure 4, the RIAs (11) and (12) and the transitivity

$$\mathsf{Mortal} \sqsubseteq \leqslant 2\,\mathsf{hasParent}.\top \tag{8}$$
$$\mathsf{Primordial} \equiv \{\mathrm{uranus}\} \sqcup \{\mathrm{gaia}\} \tag{9}$$
$$\mathsf{Deity} \sqsubseteq \mathsf{Primordial} \sqcup \exists \mathsf{hasAncestor}.\mathsf{Primordial} \tag{10}$$
$$\mathsf{hasParent}^- \equiv \mathsf{hasChild} \tag{11}$$
$$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor} \tag{12}$$
$$\mathsf{trans}(\mathsf{hasAncestor}) \tag{13}$$

Fig. 4. TBox axioms in $\mathcal{SHOIQ}$

axiom (13). The first GCI (8) in the figure illustrates the usefulness of number restrictions, which allow us to restrict the number of parents of each mortal to two. Note that this GCI combined with (1), (4), (6) and (7) actually ensures that each mortal has exactly two parents: one mother that is female and one father that is male. The GCI (9) illustrates the power of nominals: it says that the primordial gods are exactly Uranus and Gaia. The third GCI, (10), says that all non-primordial deities descend from a primordial god. The hasChild relation is exactly the inverse of hasParent, that is d has a parent d' if and only if d' has a child d (11). All parents are ancestors (12), and the ancestor relation is transitive (13). That is, the ancestors of the ancestors of d are ancestors of d, for every d.

Semantics of $\mathcal{SHOIQ}$. To give semantics to $\mathcal{SHOIQ}$ knowledge bases, we need to define the semantics of the new concept and role constructors.

Definition 11 (semantics of concepts and roles in $\mathcal{SHOIQ}$). *For every interpretation $\mathcal{I}$, we define:*

$$\begin{aligned}
(p^-)^{\mathcal{I}} &= \{(d', d) \mid (d, d') \in p^{\mathcal{I}}\} \\
\{a\}^{\mathcal{I}} &= \{a^{\mathcal{I}}\} \\
(\geqslant n\, S.C)^{\mathcal{I}} &= \{d \mid |\{d' \mid (d, d') \in S^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\}| \geq n\} \\
(\leqslant n\, S.C)^{\mathcal{I}} &= \{d \mid |\{d' \mid (d, d') \in S^{\mathcal{I}} \wedge d' \in C^{\mathcal{I}}\}| \leq n\}
\end{aligned}$$

We also need to define the semantics of assertions and axioms, on which the semantics of knowledge bases depends.

Definition 12 (satisfaction of ABox assertions and TBox axioms). *Let $\mathcal{I}$ be an interpretation. The* satisfaction *relation $\mathcal{I} \models \gamma$ if γ is an ABox assertion, a GCI or a RIA, is as for $\mathcal{ALCH}$.*

For transitivity axioms, we have that $\mathcal{I} \models \mathsf{trans}(P)$ if $P^{\mathcal{I}}$ is transitively closed, that is, if for every d_1, d_2, d_3 in $\Delta^{\mathcal{I}}$, $(d_1, d_2) \in P^{\mathcal{I}}$ and $(d_2, d_3) \in P^{\mathcal{I}}$ implies $(d_1, d_3) \in P^{\mathcal{I}}$.

Example 6. Recall the interpretation $\mathcal{I}_1$ from Example 4, and additionally let

$$\begin{aligned} \mathsf{gaia}^{\mathcal{I}_1} &= 6 \\ \mathsf{uranus}^{\mathcal{I}_1} &= 7 \\ \mathsf{hasChild}^{\mathcal{I}_1} &= \{(d', d) \in \Delta^{\mathcal{I}_1} \times \Delta^{\mathcal{I}_1} \mid (d, d') \in \mathsf{hasParent}^{\mathcal{I}_1}\}, \end{aligned}$$

and let $\mathsf{hasAncestor}^{\mathcal{I}_1}$ be the transitive closure of $\mathsf{hasParent}^{\mathcal{I}_1}$. Then $\mathcal{I}_1 \models \mathcal{K}_2^{theo}$.

Sublogics of $\mathcal{SHOIQ}$. There are many well known DLs that contain $\mathcal{ALC}$, and extend it with some of the features of $\mathcal{SHOIQ}$. The logic $\mathcal{S}$ is the extension of $\mathcal{ALC}$ with transitivity axioms. Both $\mathcal{ALC}$ and $\mathcal{S}$ can be extended with the additional features as follows: the presence of the letter $\mathcal{H}$ indicates that RIAs are allowed, and the additional letters $\mathcal{I}$, $\mathcal{O}$ and $\mathcal{Q}$ respectively denote the presence of inverses as a role constructor, of nominals, and of number restrictions. Some of these extensions, which will be mentioned throughout the chapter, are listed in Table 1. The best known of them is $\mathcal{SHIQ}$, which is closely related to the OWL-Lite standard.

Table 1. Some expressive DLs between $\mathcal{ALC}$ and $\mathcal{SHOIQ}$

DL	TAs	RIAs	inverses	nominals	NRs
$\mathcal{ALC}$					
$\mathcal{ALCI}$			✓		
$\mathcal{ALCHQ}$		✓			✓
$\mathcal{SH}$	✓	✓			
$\mathcal{SHIQ}$	✓	✓	✓		✓
$\mathcal{SHOQ}$	✓	✓		✓	✓
$\mathcal{SHOI}$	✓	✓	✓	✓	
$\mathcal{ALCHOIQ}$		✓	✓	✓	✓
$\mathcal{SHOIQ}$	✓	✓	✓	✓	✓

Example 7. Recall the knowledge base $\mathcal{K}_2^{theo} = \langle \mathcal{A}^{theo}, \mathcal{T}_2^{theo} \rangle$ from the previous example. In DLs that do not have inverses, like $\mathcal{ALCH}$ and $\mathcal{SHQ}$, the RIA (11) cannot be expressed, hence we can only use the relations $\mathsf{hasParent}$ and $\mathsf{hasChild}$ as two independent roles, and the intended relationship between them cannot be enforced. In DLs that do not support transitivity axioms, like $\mathcal{ALC}$, we can only ensure that parents of an object d are its ancestors, but we cannot relate d to, for example, the parents of its parents.

2.4 The $\mathcal{SR}$ Family

$\mathcal{SROIQ}$ is a rather well known extension of $\mathcal{SHOIQ}$, which was proposed as the basis for the Web Ontology Language standard OWL 2 [26]. Its sublogics $\mathcal{SRIQ}$, $\mathcal{SROQ}$ and $\mathcal{SROI}$ are analogous to $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIO}$.

The most prominent feature of the logics in the $\mathcal{SR}$ family are complex role inclusion axioms of the form $P_1 \circ \cdots \circ P_n \sqsubseteq P$. It is also possible to explicitly state certain properties of roles like transitivity, (ir)reflexivity and disjointness. Some of these additions increase the expressivity of the logic, while others are just 'syntactic sugar' and are intended to be useful for ontology engineering. We recall the definition of $\mathcal{SROIQ}$ from [47], borrowing some notation from [55].

As usual, we start by defining concepts and roles.

Definition 13 ($\mathcal{SROIQ}$ concepts and roles). *In $\mathcal{SROIQ}$, we assume that the signature contains a special role name U, called the* universal *role.* Atomic concepts B, concepts C, atomic roles P, S, *and* roles R, *obey the following grammar, where $a \in \mathsf{N_I}$, $A \in \mathsf{N_C}$, $p \in \mathsf{N_R}$:*

$$\begin{aligned} B &::= A \mid \{a\} \\ C, C_1, C_2 &::= B \mid \neg C \mid C_1 \sqcap C_2 \mid C_1 \sqcup C_2 \mid \forall P.C \mid \exists P.C \mid \forall P.C \mid \exists \mathsf{U}.C \mid \forall \mathsf{U}.C \mid \\ &\quad \geqslant n\, S.C \mid \leqslant n\, S.C \mid \exists S.\mathsf{Self} \\ P, S &::= p \mid p^- \\ R, R_1, R_2 &::= S \mid R_1 \circ R_2 \end{aligned}$$

We denote by $\overline{\mathsf{N_R}}$ the set of all atomic roles $\{p, p^- \mid p \in \mathsf{N_R}\}$. *Non-atomic roles of the form $P_1 \circ \cdots \circ P_n$ may be called* role chains.

Note that U may only occur in universal and existential restrictions. $\mathcal{SROIQ}$ supports some assertions and axioms that were not present in the other logics so far. In particular, the rich role axioms are its main distinguishing feature.

Definition 14 ($\mathcal{SROIQ}$ ABox assertions, TBox axioms). *In $\mathcal{SROIQ}$,* ABox assertions *are as follows:*

- *If C is a concept and $a \in \mathsf{N_I}$ an individual, then $C(a)$ is a* concept membership assertion.
- *If P is an atomic role and $a, b \in \mathsf{N_I}$ are individuals, then $P(a, b)$ is a* (positive) role membership assertion.
- *If S is an atomic role and $a, b \in \mathsf{N_I}$ are individuals, then $\neg S(a, b)$ is a* (negative) role membership assertion.
- *If $a, b \in \mathsf{N_I}$ are individuals, then $a \not\approx b$ is an* inequality assertion.

TBox axioms *are GCIs, defined as usual, as well as:*

- *If R is a role chain and P is an atomic role, then $R \sqsubseteq P$ is a* complex role inclusion axiom (CRIA).
- *If P, S, S' are atomic roles, then the following are* role property axioms:[2]

$$\mathsf{Ref}(P), \qquad \mathsf{Irr}(S), \qquad \mathsf{Asy}(S), \qquad \textit{and} \qquad \mathsf{Dis}(S, S'),$$

[2] We use the term *role property axiom* instead of *role assertions* used in [46], since the latter is often used to refer to the ABox role membership assertions.

To define $\mathcal{SROIQ}$ knowledge bases, we need some additional conditions that were designed to ensure decidability. In particular, we need a notion called *regularity* and, similarly to $\mathcal{SHOIQ}$, we must define *simple roles*, and restrict the roles occurring in certain positions to be simple. As for $\mathcal{SHOIQ}$, we define a relation $\sqsubseteq^{\mathcal{T}}$ that contains the pairs R, P of roles such that $R^{\mathcal{I}} \subseteq P^{\mathcal{I}}$ for each model $\mathcal{I}$ of $\mathcal{T}$, but the definition is more involved due to the presence of role chains in the role inclusion axioms.

Definition 15 (regular TBoxes, simple roles, knowledge bases). *A TBox $\mathcal{T}$ is* regular*, if there exists a strict partial order $\prec$ on the set $\overline{\mathsf{N_R}}$ of all atomic roles such that $\mathsf{Inv}(P) \prec P'$ if and only if $P \prec P'$ for every $P, P' \in \overline{\mathsf{N_R}}$, and such that every CRIA in $\mathcal{T}$ is of one of the following forms:*

- $P \circ P \sqsubseteq P$,
- $P^- \sqsubseteq P$,
- $R \sqsubseteq P$,
- $R \circ P \sqsubseteq P$, *or*
- $P \circ R \sqsubseteq P$,

where $R = P_1 \circ \cdots \circ P_n$ and $P_i \prec P$ for each $1 \leq i \leq n$.

The relation $\sqsubseteq^{\mathcal{T}}$ is the smallest relation such that

(i) $P \sqsubseteq^{\mathcal{T}} P$ for every role $P \in \overline{\mathsf{N_R}}$ such that P or $\mathsf{Inv}(P)$ occurs in $\mathcal{T}$, and

(ii) $P_1 \circ \cdots \circ P_n \sqsubseteq^{\mathcal{T}} P$ for each $P_1 \circ \cdots \circ P_{i-1} \circ P' \circ P_{j+1} \circ \cdots \circ P_n \sqsubseteq^{\mathcal{T}} P$ such that $P_i \circ \cdots \circ P_j \sqsubseteq P' \in \mathcal{T}$ or $\mathsf{Inv}(P_j) \circ \cdots \circ \mathsf{Inv}(P_i) \sqsubseteq P' \in \mathcal{T}$, for some $P' \in \overline{\mathsf{N_R}}$ and $1 \leq i \leq j \leq n$.

A role P is simple *w.r.t. $\mathcal{T}$, if $P \notin \{\mathsf{U}, \mathsf{U}^-\}$ and there are no roles $P_1, \ldots, P_n$ with $n \geq 2$ such that $P_1 \circ \cdots \circ P_n \sqsubseteq^{\mathcal{T}} P$.*

A $\mathcal{SROIQ}$ knowledge base *is a triple $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ where $\mathcal{A}$ in an ABox, $\mathcal{T}$ is a* regular *TBox, and additionally all roles S, S' occurring in the following kinds of expressions are simple w.r.t. $\mathcal{T}$:*

- *number restrictions $\geqslant n\, S.C$, $\leqslant n\, S.C$,*
- *concepts of the form $\exists S.\mathsf{Self}$,*
- *negative role assertions $\neg S(a, b)$ in $\mathcal{A}$, and*
- *role property axioms $\mathsf{Irr}(S)$, $\mathsf{Asy}(S)$ or $\mathsf{Dis}(S, S')$ in $\mathcal{T}$.*

Semantics of $\mathcal{SROIQ}$. We only need to give semantics to the new concept constructor $\exists S.\mathsf{Self}$ and the new role constructor $R \circ R'$, as well as to the special ABox assertions and TBox axioms.

Definition 16 (semantics of $\mathcal{SHOIQ}$ concepts and roles) *For every interpretation $\mathcal{I}$, we define:*

$$\begin{aligned} (\exists S.\mathsf{Self})^{\mathcal{I}} &= \{d \mid (d, d) \in S^{\mathcal{I}}\} \\ (R \circ R')^{\mathcal{I}} &= R^{\mathcal{I}} \circ R'^{\mathcal{I}} \end{aligned}$$

Here, we override the $\circ$ operator to denote the composition of two binary relations. That is, for binary relations rel_1 and rel_2 with $rel_i \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, we define $rel_1 \circ rel_2 = \{(d_1, d_3) \mid \exists d_2 \in \Delta^{\mathcal{I}} : (d_1, d_2) \in rel_1 \text{ and } (d_2, d_3) \in rel_2\}$.

$$\mathsf{hasAncestor} \circ \mathsf{hasAncestor} \sqsubseteq \mathsf{hasAncestor} \tag{14}$$

$$\{\text{narcissus}\} \sqsubseteq \exists \mathsf{isInLoveWith.Self} \tag{15}$$

$$\exists \mathsf{hasParent.Self} \sqsubseteq \bot \tag{16}$$

$$\mathsf{hasParent} \sqsubseteq \mathsf{hasDirectRelative} \tag{17}$$

$$\mathsf{hasChild} \sqsubseteq \mathsf{hasDirectRelative} \tag{18}$$

$$\mathsf{hasParent} \circ \mathsf{hasParent}^{-} \sqsubseteq \mathsf{hasSibling} \tag{19}$$

$$\mathsf{hasParent} \circ \mathsf{hasSibling} \circ \mathsf{hasChild} \sqsubseteq \mathsf{hasCousin} \tag{20}$$

$$\mathsf{Dis}(\mathsf{hasParent}, \mathsf{hasChild}) \tag{21}$$

Fig. 5. TBox axioms (15)–(21) in $\mathcal{SROIQ}$

Definition 17 (satisfaction of ABox assertions and TBox axioms). *Let $\mathcal{I}$ be an interpretation. Then $\mathcal{I}$ satisfies a negated role membership assertion $\neg S(a,b)$, i.e., $\mathcal{I} \models \neg S(a,b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin S^{\mathcal{I}}$. For other ABox assertions, all TBox axioms, and CRIAs, satisfaction is defined as usual.*

For role property axioms, we have:

- $\mathcal{I} \models \mathsf{Ref}(P)$ *if* $R^{\mathcal{I}}$ *is* reflexive*, i.e.,* $(d,d) \in P^{\mathcal{I}}$ *for every* $d \in \Delta^{\mathcal{I}}$*;*
- $\mathcal{I} \models \mathsf{Irr}(S)$ *if* $S^{\mathcal{I}}$ *is* irreflexive*, i.e.,* $(d,d) \notin S^{\mathcal{I}}$ *for every* $d \in \Delta^{\mathcal{I}}$*;*
- $\mathcal{I} \models \mathsf{Asy}(S)$ *if* $S^{\mathcal{I}}$ *is* asymmetric*, i.e.,* $(d,d') \in S^{\mathcal{I}}$ *implies* $(d',d) \notin S^{\mathcal{I}}$*; and*
- $\mathcal{I} \models \mathsf{Dis}(S,S')$ *if the relations* S *and* S' *are* disjoint*, i.e.,* $S^{\mathcal{I}} \cap S'^{\mathcal{I}} = \emptyset$*.*

Example 8. Our example $\mathcal{SROIQ}$ knowledge base $\mathcal{K}^{theo} = \langle \mathcal{A}^{theo}, \mathcal{T}_4^{theo} \rangle$ has the usual ABox $\mathcal{A}^{theo}$. The TBox $\mathcal{T}_4^{theo}$ contains most of the TBox axioms from the previous examples: all the $\mathcal{ALCH}$ axioms in $\mathcal{T}_1^{theo}$ (Figure 2), and all the $\mathcal{SHOIQ}$ axioms in $\mathcal{T}_2^{theo}$ (Figure 4) except for the CRIA (13), which is replaced by the equivalent (14). Additionally it contains the $\mathcal{SROIQ}$ axioms (15)–(21) in Figure 5.

In $\mathcal{SROIQ}$, we can use Self concepts to express (15) that Narcissus is in love with himself, and (16) that no one can be his own parent. The two CRIAs (17) state that parents and children are direct relatives. The CRIAs (19) and (14) illustrate how we can express in $\mathcal{SROIQ}$ relatively complex relations, e.g., that certain relatives are siblings or cousins. Finally, the role property axiom (21) says that somebody's child cannot be also his parent.

Sublogics of $\mathcal{SROIQ}$. We consider three sublogics of $\mathcal{SROIQ}$ which are analogous to $\mathcal{SHIQ}$, $\mathcal{SHOQ}$ and $\mathcal{SHIO}$.

Definition 18 (The DLs $\mathcal{SRIQ}$, $\mathcal{SROQ}$ and $\mathcal{SROI}$). *A knowledge base $\mathcal{K}$ in $\mathcal{SROIQ}$ is*

- *in* $\mathcal{SRIQ}$*, if no* nominal concepts $\{a\}$ *occur;*
- *in* $\mathcal{SROQ}$*, if no* inverse roles P^- *occur; and*
- *in* $\mathcal{SROI}$*, if no* number restrictions $\geqslant n\, S.C$, $\leqslant n\, S.C$ *occur.*

We remark that the definition of $\mathcal{SROIQ}$ presented here is a minor restriction of the original definition in [47], but it is not less expressive. In particular, Horrocks et al. allow role property axioms of two additional forms. First, they allow transitivity axioms $\mathsf{trans}(P)$, as in $\mathcal{SHOIQ}$, which can be equivalently expressed using CRIAs of the form $P \circ P \sqsubseteq P$. Second, $\mathsf{Sym}(P)$ asserts that P is symmetric, which can be expressed as $P^- \sqsubseteq P$; please note that this is not supported in the $\mathcal{SROQ}$ fragment. ABox assertions $\neg R(a,b)$ for non-simple R are allowed in $\mathcal{SROIQ}$ [47], but not in $\mathcal{SRIQ}$ [46]. Our syntax allows us to express them in all logics with nominals, since the assertion $\neg R(a,b)$ can be equivalently rewritten as $\{a\} \sqsubseteq \forall R.\neg\{b\}$.

To allow for a more uniform definition of $\mathcal{SROIQ}$ and its sublogics, we have only allowed for the universal role in universal and existential restrictions. The definition of $\mathcal{SROIQ}$ in [47] allows U to occur as an ordinary role everywhere except in CRIAs and role property axioms, while the definition of $\mathcal{SRIQ}$ in [46] does not allow it. Our $\mathcal{SROIQ}$ definition is not less expressive, as role membership assertions $\mathsf{U}(a,b)$ or $\mathsf{U}^-(a,b)$ are trivially satisfied in every interpretation and can be ignored, while $\neg\mathsf{U}(a,b)$ or $\neg\mathsf{U}^-(a,b)$ are trivially unsatisfiable and can be replaced by $\bot(a)$. Concepts of the form $\exists\mathsf{U}.\mathsf{Self}$ and $\exists\mathsf{U}^-.\mathsf{Self}$ are equivalent to $\top$. Hence, the only interesting consequence of our restrictions on the occurrences of the universal role are that one cannot write number restrictions $\leqslant n\, \mathsf{U}.C$, $\geqslant n\, \mathsf{U}.C$ in our syntax. This is not a limitation, since they can be simulated using nominals. To simulate the effect of $\leqslant n\, \mathsf{U}.C$, we replace it by a fresh concept name $A_{(\leqslant n\, \mathsf{U}.C)}$ wherever it occurs in $\mathcal{K}$, and make sure that whenever an object satisfies $A_{(\leqslant n\, \mathsf{U}.C)}$ in a model, then the cardinality of the interpretation of C is bounded by n. The latter can be achieved using n fresh individuals $a_1, \ldots, a_n$, and a GCI

$$\exists\mathsf{U}.A_{(\leqslant n\, \mathsf{U}.C)} \sqcap C \sqsubseteq \{a_1\} \sqcup \cdots \sqcup \{a_n\},$$

which ensures that if the interpretation of $A_{(\leqslant n\, \mathsf{U}.C)}$ is non-empty, then there are at most n instances of C. Dually, $\geqslant n\, \mathsf{U}.C$ means that in every model the cardinality of the interpretation of C has to be at least n. We also replace $\geqslant n\, \mathsf{U}.C$ by a fresh concept name $A_{(\geqslant n\, \mathsf{U}.C)}$. To put n different fresh individuals into the extension of C whenever the interpretation of $A_{(\geqslant n\, \mathsf{U}.C)}$ is not empty, we use a GCI

$$A_{(\geqslant n\, \mathsf{U}.C)} \sqsubseteq \prod_{1 \leq i \leq n} \exists\mathsf{U}.(C \sqcap \{a_i\} \sqcap \prod_{1 \leq j < n} \neg\{a_j\}).$$

Note that these number restrictions are not supported by $\mathcal{SRIQ}$ according to [46], and they cannot be added to it without modifying its expressive power and computational complexity. Allowing expressions of the form $\leqslant n\, \mathsf{U}.C$ amounts to imposing a cardinality restriction on the (interpretation of) the concept C. This is enough to simulate nominals [94], making $\mathcal{SRIQ}$ as expressive as $\mathcal{SROIQ}$.

3 Reasoning in DLs

As knowledge representation formalisms, DLs provide reasoning services. In different application domains, there may be different kinds of implicit knowledge one would like to infer from the knowledge explicitely represented in the knowledge base. These different kinds of inferences are formalized as *reasoning problems*, so that algorithms for solving them can be developed and implemented. There are a few such problems that are considered important and that have been studied for most DLs, and are usually called *standard reasoning problems* (as opposed to *non-standard* problems such as reasoning with queries, which will be mentioned later).

3.1 Reasoning about Concepts

One of the most classical DL reasoning problems is *concept subsumption*, that is, given two (complex) concepts C and D, we want to decide whether the interpretation of C is always a subset of the interpretation of D, no matter how the symbols in the vocabulary are interpreted. Intuitively, this means that concept D is more general than concept C. Another important problem is to decide whether a concept is satisfiable, that is, whether it does not contain any contradiction and has a non-empty extension in some interpretation.

Concept Subsumption: Given two concepts C and D, decide whether C *is subsumed by* D, that is, whether $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every interpretation $\mathcal{I}$.

Concept Satisfiability: Given a concept C, decide whether C *is satisfiable*, that is, whether there exists an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

These problems are closely related. In all description logics where the bottom concept $\bot$ is allowed (or can be simulated, for example, using a concept of the form $A \sqcap \neg A$), concept unsatisfiability can be reduced to subsumption: C is unsatisfiable if and only if C is subsumed by $\bot$. Similarly, subsumption can be reduced to unsatisfiability whenever $\sqcap$ and $\neg$ are available, since C is subsumed by D if and only if $C \sqcap \neg D$ is unsatisfiable.

Example 9. Suppose that we want to define some complex concept 'Parent of Heroes' (PoH) to use in our ontology, which should define a class of mortals and deities whose sons are heroes. We come up with the following two possible concepts. We want to know whether they are satisfiable, since this may help us realize if we have made some mistake:

$$\mathsf{PoH}_1 = \mathsf{Mortal} \sqcap \mathsf{Deity} \sqcap \forall \mathsf{hasChild}.(\mathsf{Hero} \sqcap \mathsf{Male} \sqcap \neg \mathsf{Hero})$$
$$\mathsf{PoH}_2 = (\mathsf{Mortal} \sqcup \mathsf{Deity}) \sqcap \exists \mathsf{hasChild}.(\mathsf{Hero} \sqcap \mathsf{Male}) \sqcap \forall \mathsf{hasChild}.(\mathsf{Hero} \sqcap \neg \mathsf{Male})$$

The first concept PoH_1 is *satisfiable*: it is satisfied in an interpretation where there is some individual that is simultaneously a mortal and a deity, and this individual has no children. Note that this individual would not satisfy the concept if it had any children, since all its children would have to be at the same

time hero and not hero, which is a contradiction. Hence this concept, although satisfiable, may not reflect properly the intended meaning of our PoH concept.

The second concept PoH_1 is *unsatisfiable*: for an individual to satisfy that concept it has to simultaneously satisfy the following: (1) it must be either a mortal or a deity, which is possible, (2) it must have a male child that is a hero, which is also possible, (3) all its children must be heroes and not males; this is possible on its own, but it contradicts (2). Hence, in every interpretation, the extension of PoH_2 is empty.

We use what we have learned from these two concepts to improve our definitions of PoH, and come up with the following two improved versions, which are both satisfiable:

$$\mathsf{PoH}_3 = (\mathsf{Mortal} \sqcup \mathsf{Deity}) \sqcap \exists \mathsf{hasChild}.(\top) \sqcap \forall \mathsf{hasChild}.(\mathsf{Male} \sqcap \mathsf{Hero})$$
$$\mathsf{PoH}_4 = (\mathsf{Mortal} \sqcup \mathsf{Deity}) \sqcap \exists \mathsf{hasChild}.(\mathsf{Male}) \sqcap \forall \mathsf{hasChild}.(\neg\mathsf{Male} \sqcup \mathsf{Hero})$$

We would like to know which of the two definitions is more general, hence we test for mutual subsumption, and we see that PoH_3 is subsumed by PoH_4, but not vice-versa. Both of them require objects to be either mortals or deities, and to have at least one son. The difference is that in PoH_3 all children are required to be male and heroes, while in PoH_4 all sons must be heroes, but no constraint is imposed on the children that are not male. For example, a mortal that has a son that is a hero, but also has a daughter, would satisfy PoH_4, but not PoH_3.

3.2 Reasoning about KBs

So far we have only considered reasoning about concepts, but it is also possible, for example, to consider the above problems w.r.t. a given KB or a TBox. We say that C is subsumed by D w.r.t. a KB $\mathcal{K}$ (resp., a TBox $\mathcal{T}$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{K}$ (resp., of $\mathcal{T}$); and that C is satisfiable w.r.t. $\mathcal{K}$ (resp., $\mathcal{T}$) if there exists a model $\mathcal{I}$ of $\mathcal{K}$ (resp., $\mathcal{T}$) such that $C^{\mathcal{I}} \neq \emptyset$.

Example 10. Recall the concept PoH_1 from the example above. It is satisfiable on its own, but unsatisfiable w.r.t. any TBox containing the GCI $\mathsf{Deity} \sqsubseteq \neg\mathsf{Mortal}$ stating that deities are not mortal. It is also unsatisfiable w.r.t. any KB containing such a TBox.

TBox axioms may also have an effect on subsumption. Recall from the previous example that PoH_3 is subsumed by PoH_4, but not vice-versa. If we consider a (rather unusual) GCI $\mathsf{Mortal} \sqcup \mathsf{Deity} \sqsubseteq \forall \mathsf{hasChild}.\mathsf{Male}$ stating that mortals and deities can only have male children, then PoH_4 would be subsumed by PoH_3, and both concepts would be equivalent.

As for the case of reasoning about concepts only, concept satisfiability and subsumption w.r.t. a KB are interreducible: C is unsatisfiable w.r.t. $\mathcal{K}$ if and only if C is subsumed by $\bot$ w.r.t. $\mathcal{K}$, and C is subsumed by D w.r.t. $\mathcal{K}$ if and only if $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{K}$.

In the problems we have discussed so far, the TBox plays a more prominent role than the ABox. Now we introduce a reasoning problem where the assertional information in the ABox plays a crucial role.

Instance Checking. Given a concept C, an individual a, and a knowledge base $\mathcal{K}$, decide whether *a is an instance of C w.r.t. $\mathcal{K}$*, that is, whether $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$.

$$\begin{aligned}
\mathsf{Deity}\,&(\text{zeus})\\
\mathsf{Mortal}\,&(\text{alcmene})\\
\mathsf{Male}\,&(\text{heracles})\\
\mathsf{Male}\,&(\text{perseus})\\
\mathsf{hasMother}\,&(\text{heracles}, \text{alcmene})\\
\mathsf{hasFather}\,&(\text{heracles}, \text{zeus})\\
\mathsf{hasMother}\,&(\text{perseus}, \text{zeus})
\end{aligned}$$

(a) ABox

$$\begin{aligned}
\mathsf{Mortal} &\sqsubseteq \neg\mathsf{Deity}\\
\mathsf{Male} \sqcap \exists\mathsf{hasParent}.\mathsf{Deity} &\sqsubseteq \mathsf{Hero}\\
\mathsf{hasMother} &\sqsubseteq \mathsf{hasParent}\\
\mathsf{hasFather} &\sqsubseteq \mathsf{hasParent}\\
\mathsf{hasChild} &\equiv \mathsf{hasParent}^{-}
\end{aligned}$$

(b) TBox

Fig. 6. Knowledge Base $\mathcal{K}_{ic}$ for Example 6

Example 11. Consider the KB $\mathcal{K}_{ic}$ in Figure 6 and the concept $\mathsf{PoH}_4 = (\mathsf{Mortal} \sqcup \mathsf{Deity}) \sqcap \exists\mathsf{hasChild}.(\mathsf{Male}) \sqcap \forall\mathsf{hasChild}.(\neg\mathsf{Male} \sqcup \mathsf{Hero})$ from the previous examples.

First we observe that Zeus is an instance of the concept PoH_4 w.r.t. to $\mathcal{K}_{ic}$, since he is a deity, has at least one male child (in fact, at least two: Heracles and Perseus), and the second axiom in the TBox implies that every male child of Zeus is a hero.

In contrast, Alcmene is not an instance of PoH_4 w.r.t. $\mathcal{K}_{ic}$. She clearly satisfies the first two conjuncts, since she is mortal and has a male child, Heracles. However, she does not satisfy the third conjunct in every model of $\mathcal{K}_{ic}$. Although we know that Heracles is a hero, in arbitrary models of $\mathcal{K}_{ic}$ she can have additional male children who may not be heroes. Note that Alcmene would become an instance of PoH_4 is we add to $\mathcal{K}_{ic}$ an assertion $\leqslant 1\,\mathsf{hasChild}.\mathsf{Male}(\text{alcmene})$ stating that Alcmene has only one son.

Next we introduce *knowledge base satisfiability*, which informally consists on deciding whether a knowledge base representing some domain knowledge does not contain contradictions. Formally, we have:

Definition 19 (knowledge base satisfiability). *A knowledge base $\mathcal{K}$ is* satisfiable *if there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$, and* unsatisfiable *otherwise. The* knowledge base satisfiability problem *is to decide whether a given knowledge base is satisfiable or unsatisfiable.*

Example 12. The KB $\mathcal{K}_{ic}$ in Figure 6 is satisfiable, but would become unsatisfiable if we added to the ABox the fact $\neg\mathsf{PoH}_4(\text{zeus})$.

This problem is of interest on its own, but it is also central because many other reasoning problems can be reduced to it, and we can thus reuse knowledge base satisfiability algorithms to provide other reasoning services. In fact, most of the state-of-the art reasoners for expressive DLs implement KB satisfiability algorithms and use them to provide a range of reasoning services. Instance checking can be reduced to knowledge base satisfiability by adding to the ABox the negation of the assertion we want to test, i.e., a is an instance of C w.r.t. $\mathcal{K} = \langle \mathcal{A}, \mathcal{T} \rangle$ if and only if $\langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T} \rangle$ is unsatisfiable. Concept unsatisfiability (and hence concept subsumption) can be tested similarly: C is unsatisfiable w.r.t. $\mathcal{K}$ if and only if $\langle \mathcal{A} \cup \{\neg C(a)\}, \mathcal{T} \rangle$ is unsatisfiable, where a is a fresh individual name not occurring in $\mathcal{K}$.

Reasoning about Concepts vs. Reasoning about TBoxes. We have shown that the main problems for reasoning about concepts reduce easily to reasoning about TBoxes and KBs. The converse is not always the case and in many DLs, reasoning about concepts is easier than reasoning about TBoxes and KBs [70]. For example, for $\mathcal{ALC}$, concept satisfiability is a PSPACE-complete problem, while reasoning w.r.t. to arbitrary TBoxes or KBs is EXPTIME hard [92,70]. The upper bounds for reasoning about concepts can sometimes be extended to TBoxes and KBs that satisfy some *acyclicity conditions* [64]. In some expressive DLs, in particular in $\mathcal{SH}$ and its extensions, reasoning about concepts only is not easier anymore. Using a technique known as *internalization* [5,92], any TBox can be rewritten into an equivalent concept, and reasoning about one concept only is already as hard as reasoning w.r.t. arbitrary TBoxes, namely EXPTIME-hard.

The ABox and its Effect on Reasoning. In DLs that support nominals (i.e., in those whose name contains $\mathcal{O}$), ABox assertions can be equivalently expressed as GCIs: $C(a)$ is equivalent to $\{a\} \sqsubseteq C$ and $R(a, b)$ is equivalent to $\{a\} \sqsubseteq \exists R.\{b\}$. Hence, for these logics, reasoning about full KBs and reasoning about TBoxes only are not different. In DLs without nominals, in contrast, the ABox provides additional expressivity and calls for more involved reasoning algorithms, although for expressive DLs, it usually has no impact on the worst case complexity of reasoning, cf. [50].

4 Ontologies and Data Access

The traditional setting of DLs that we have discussed so far can be considered a relatively mature field. The computational complexity of the main reasoning problems is well understood for most logics, and moreover, efficient algorithms have been developed and implemented. Existing reasoners are highly optimized and can handle large knowledge bases, even for logics of high computational complexity.

DLs are often deployed in application areas where there is data involved, which may be stored in a traditional relational database, an XML or RDF dataset, etc. The data can usually be seen as an ABox, and it is then very natural to have an

ontology (or TBox) describing the organization of the data. The ontology can provide a high-level conceptual view of the data repository, and specify implicit concepts and roles that extend the vocabulary of the database with terms that are relevant for specific applications. The ontology can then be exploited to access the data and to answer queries taking into account the knowledge that is implicit in the ontology.

$\mathsf{Deity}(\text{zeus})$	$\mathsf{Deity} \sqsubseteq \mathsf{Primordial} \sqcup \exists \mathsf{hasAncestor}.\mathsf{Primordial}$
$\mathsf{Primordial}(\text{gaia})$	$\mathsf{Primordial} \sqsubseteq \mathsf{Deity}$
$\mathsf{hasMother}(\text{zeus}, \text{rhea})$	$\exists \mathsf{hasParent}.\top \sqsubseteq \neg \mathsf{Primordial}$
$\mathsf{hasMother}(\text{chronos}, \text{gaia})$	$\mathsf{hasMother} \sqsubseteq \mathsf{hasParent}$
	$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}$
(a) ABox	(b) TBox

Fig. 7. Knowledge Base for Example 13

Example 13. We want to pose over the data given by the ABox in Figure 7a a query that retrieves all individuals that have a divine ancestor. The query can be expressed with the following formula:

$$q_1^{theo}(x) : -\exists v.\mathsf{hasAncestor}(x, v) \wedge \mathsf{Deity}(v)$$

If we see the ABox as a plain database, then the query has no answers since there are no facts of the form $\mathsf{hasAncestor}(a, b)$. However, the TBox in 7b gives implicit knowledge about the data that can be used to infer the complete answers. For example, the last two axioms allow us to infer that Gaia is an ancestor of Chronos, and by the second axiom Gaia is a deity, hence Chronos is an answer to the query. Furthermore, by the third axiom, Zeus is not a primordial deity, and the second axiom implies that it has some ancestor that is a primordial deity. This implies that Zeus is also in the query answer, even though we do not know who his primordial ancestor is.

This kind of query answering over knowledge bases has become a vibrant field of research known as *Ontology Based Data Access (OBDA)*. In the rest of the chapter we give a fast tour through the field, the problems that it aims to solve, the techniques that are used, and the main results obtained so far. Following the common practices of OBDA, in this section we will often use the term *data* to refer to ABoxes, and *ontology* to refer to TBoxes.

4.1 Choosing a Query Language

The first question that arises in the OBDA setting, is what kinds of queries we want to pose.

Instance Queries. The first natural language that comes to mind is to use DL concepts as a query language. Some interesting queries can be expressed as a concept C, and then each individual that is an instance of C is called an answer to C. This kind of queries are known as *instance queries*.

Example 14. The query $q_1^{theo}(x)$ above that retrieves all individuals that have a divine ancestor can also be expressed using the following concept:

$$\exists \mathsf{hasAncestor}.\mathsf{Deity}$$

Answering instance queries trivially reduces to the instance checking problem, hence it is often considered an standard reasoning task.

Unfortunately, the expressivity of instance queries is rather limited, and many natural queries are not expressible as instance queries. For example, we can use instance queries to find individuals that have a divine ancestor, are not deities themselves, and have a child who has fathered only heroes. However, we cannot relate the different objects to find pairs of individuals that share an ancestor, or that have such a child together. It is also not possible to find cycles of direct relatives that have certain properties. A good query language should support some means for combining pieces of information, for example using variables.

First-Order Queries. A natural alternative would be to use as a query language FOL, which is more expressive than DLs. This would allow us to express quite complex queries, including the ones above.

Example 15. The following is a FOL query:

$$\begin{aligned} q_2^{theo}(x_1, x_2) = \; & \mathsf{Male}(x_1) \wedge \mathsf{Female}(x_2) \wedge (\neg \mathsf{Deity}(x_1) \vee \neg \mathsf{Deity}(x_2)) \wedge \\ & \exists v_1 . \mathsf{hasAncestor}(x_1, v_1) \wedge \mathsf{hasAncestor}(x_2, v_1) \wedge \\ & \exists v_2 . \big(\mathsf{hasChild}(x_1, v_2) \wedge \mathsf{hasChild}(x_2, v_2) \wedge \\ & \forall v_3 . \mathsf{hasChild}(v_2, v_3) \rightarrow \mathsf{Hero}(v_3)\big) \end{aligned}$$

Informally, it asks for pairs of individuals x_1 and x_2, where one is male and the other female and at least one of them is not a deity, who are relatives (i.e., they have a common ancestor v_1), and have a common child v_2 all of whose children are heroes.

Another FOL query is

$$\begin{aligned} q_3^{theo} = \exists v_1, v_2, v_3, v_4 . \; & \mathsf{hasDirectRelative}(v_1, v_2) \wedge \mathsf{hasDirectRelative}(v_2, v_3) \wedge \\ & \mathsf{hasDirectRelative}(v_3, v_4) \wedge \mathsf{hasDirectRelative}(v_4, v_1) \wedge \\ & \mathsf{Deity}(v_1) \wedge \mathsf{Hero}(v_2) \wedge \mathsf{Male}(v_4) \end{aligned}$$

This query returns only **true** or **false** as an answer. It asks whether there is a cycle of four direct relatives v_1, v_2, v_3, v_4 such that v_1 is a deity, v_2 is a hero, and v_4 is male.

FOL would make a powerful query language, but it has a major drawback. Deciding whether some object is in the certain answers of a query is an undecidable

problem, since it basically amounts to deciding logical implication in FOL. For this reason, most work on query answering in DLs has considered fragments of FOL that allow to express many natural queries but still exhibit good computational properties.

Conjunctive Queries and Their Extensions. A very well known fragment of FOL that makes a convenient query language is *Conjunctive Queries (CQs)* [2]. They are the formal counterpart of the most widely used fragment of SQL (or relational algebra) queries, namely plain select-project-join queries. Disjunctions of these queries are known as *Unions of Conjunctive Queries (UCQs)*, and are also popular . Both languages have been widely studied in databases [24,2], and due to a good trade-off between expressivity and complexity, they have been adopted as core query languages in several contexts, such as query optimization [44,3], data integration [61,96,72], and ontology-based data access [83]. Most of the work concerning query answering in DLs refers in fact to CQs, and they are considered the core query languages for OBDA.

Example 16. The queries q_1^{theo} and q_3^{theo} above are both CQs, while q_2^{theo} cannot be expressed as a CQ. Another CQ is:

$$q_4^{theo}(x) = \exists v_1, v_2.\ \mathsf{Hero}(x) \wedge \mathsf{hasMother}(x, v_1) \wedge \mathsf{hasAncestor}(v_1, v_2) \wedge \mathsf{Deity}(v_2)$$

Its answers are the heroes x that have a divine ancestor on the maternal side. Finally, we give an example of a UCQ:

$$\begin{aligned} q_5^{theo}(x_1, x_2) = \ & \big(\mathsf{Male}(x_1) \wedge \mathsf{Female}(x_2) \wedge \neg\mathsf{Deity}(x_1) \wedge \\ & \exists v_1.\mathsf{hasAncestor}(x_1, v_1) \wedge \mathsf{hasAncestor}(x_2, v_1) \wedge \\ & \exists v_2.\mathsf{hasChild}(x_1, v_2) \wedge \mathsf{hasChild}(x_2, v_2)\big) \quad \vee \\ & \big(\mathsf{Male}(x_1) \wedge \mathsf{Female}(x_2) \wedge \neg\mathsf{Deity}(x_2) \wedge \\ & \exists v_1.\mathsf{hasAncestor}(x_1, v_1) \wedge \mathsf{hasAncestor}(x_2, v_1) \wedge \\ & \exists v_2.\mathsf{hasChild}(x_1, v_2) \wedge \mathsf{hasChild}(x_2, v_2)\big) \end{aligned}$$

It is similar to q_2^{theo} above, but it does not require all children of the common child v_2 of x_1 and x_2 to be heroes. Note that it is also less succinct than q_2^{theo}: since disjunctions cannot be nested inside conjunctions, we must repeat a long conjunction of atoms to express that one of x_1 and x_2 is not a deity.

In this chapter we focus on CQs. Results can often be generalized to UCQs, and sometimes even to full positive existential FOL (cf. [75]), but we will not discuss such generalizations.

Other Query Languages. For many application areas, even full FOL does not provide sufficient expressivity as a query language, and languages with higher order features such as recursion and fixed-point operators are required. A major family of such query languages are Datalog and its extensions, which we do not discuss here since they are discussed in the tutorial *Datalog and Its Extensions*

for the Semantic Web of this summer school [41]. Another notable example of queries with recursion are *regular path queries* (RPQs) [12,1,18] and their extensions, which allow one to ask for pairs of objects that are connected by a path conforming to a regular expression. They are the fundamental language for querying graph databases and data on the web. They have been studied for expressive DLs [23,22], and more recently also for lightweight DLs [10], but these languages are out of the scope of the chapter.

5 The DL Query Answering Problem

In this section, we define formally the DL Query Answering Problem. We focus on *conjunctive queries* (CQs), since they are the most popular language for query answering with ontologies. We first define the syntax and semantics of queries, and then define the *query answering* and *query entailment* problems.

5.1 Syntax and Semantics of Queries

A query atom is built from a DL concept or a DL role and is syntactically similar to an ABox assertion, but it allows for variables. These variables may be shared by the different query atoms of a query in an unrestricted way, allowing us to flexibly join pieces of information.

Definition 20 (query atom, CQs, UCQs). *Let* $\mathsf{N_V}$ *denote a countably infinite set of* variables. Query atoms *are of two forms:* concept atoms $C(v)$*, and* role atoms $R(v, v')$*, where each of the* arguments v *and* v' *is either a variable from* $\mathsf{N_V}$ *or an individual from* $\mathsf{N_I}$*,* C *is a concept, and* R *is a role.*

A conjunctive query (with answer variables $\boldsymbol{x}$) *is an expression* $\exists \boldsymbol{v}.\varphi(\boldsymbol{x}, \boldsymbol{v})$*, where* φ *is a conjunction of query atoms where only variables from* $\boldsymbol{v} \cup \boldsymbol{x}$ *occur. If* $\boldsymbol{x} = x_1, \ldots, x_n$*, then we may call* q *a query with* n *answer variables. If* $\boldsymbol{x} = \emptyset$*, that is, if all variables in* φ *are existentially quantified, then we call* q *a* Boolean *query.*

A UCQ is a disjunction of CQs, that is, a query $q = \exists \boldsymbol{v}.\big(\varphi_1(\boldsymbol{x}, \boldsymbol{v}) \vee \cdots \vee \varphi_n(\boldsymbol{x}, \boldsymbol{v})\big)$ *where each* $\varphi_i(\boldsymbol{x}, \boldsymbol{v})$ *is a conjunction of atoms.*

For a query $q = \exists \boldsymbol{v}.\varphi(\boldsymbol{x}, \boldsymbol{v})$, let $\mathsf{Atoms}(q)$ denote the set of all atoms occurring in q, and let $\mathsf{VI}(q)$ denote the set of individuals and variables that occur as arguments in its atoms.

Since a CQ $q = \exists \boldsymbol{v}.\varphi(\boldsymbol{x}, \boldsymbol{v})$ contains only conjunctions, the query is uniquely determined by its answer variables and the set of its atoms, and we do not need to store the formula φ. Further, if the query is Boolean, we can simply use the set $\mathsf{Atoms}(q)$ to represent it. Similarly, a Boolean UCQ can be represented as a collection of sets of atoms. For a Boolean UCQ $Q = \exists \boldsymbol{v}.\varphi_1 \vee \cdots \vee \varphi_n$, we may write $Q = \bigcup_{1 \leq i \leq n} \{\mathsf{Atoms}(\exists \boldsymbol{v}.\varphi_i)\}$ and, abusing notation, use $q_i \in Q$ in the place of $\mathsf{Atoms}(q_i) \in Q$ for each $q_i = \exists \boldsymbol{v}.\varphi_i$, $1 \leq i \leq n$.

Example 17. We have seen the CQs q_1^{theo}, q_3^{theo} and q_4^{theo}, and the UCQ q_5^{theo} in the examples of the previous section. The queries q_1^{theo} and q_4^{theo} both have one answer variable, x; the UCQ q_5^{theo} has two answer variables, x_1 and x_2. The query q_3^{theo} is Boolean. We give one more example of a Boolean CQ:

$$q_6^{theo} = \exists v_1, v_2, v_3, v_4 . \mathsf{Hero}(v_1) \wedge \mathsf{hasMother}(v_1, v_2) \wedge \mathsf{hasParent}(v_2, v_3) \wedge \mathsf{Deity}(v_3) \wedge \mathsf{hasChild}(v_4, v_3) \wedge \mathsf{Primordial}(v_4)$$

Informally, q_6^{theo} asks whether there exist some hero that has a divine maternal grandparent that is a child of a primordial god.

The semantics of queries is given in terms of interpretations in the natural way.

Definition 21 (binding, query match). *A* binding for a query q in an interpretation $\mathcal{I}$ *is a total function* $\pi : \mathsf{Vl}(q) \to \Delta^{\mathcal{I}}$ *such that* $\pi(a) = a^{\mathcal{I}}$ *for each individual* $a \in \mathsf{Vl}(q)$. *We write* $\mathcal{I}, \pi \models C(v)$, *if* $\pi(v) \in C^{\mathcal{I}}$, *and* $\mathcal{I}, \pi \models R(v, v')$, *if* $(\pi(v), \pi(v')) \in R^{\mathcal{I}}$.

We write $\mathcal{I}, \pi \models q$ *and call* π *a* match for q in $\mathcal{I}$ *if it makes the query true. If* q *is a CQ, this is the case if it makes all the atoms of* q *true, i.e.,* $\mathcal{I}, \pi \models \alpha$ *for all* $\alpha \in \mathsf{Atoms}(q)$. *If* $q = \bigcup_{1 \leq i \leq n} q_i$ *is a UCQ, this is the case if* π *makes some CQ in it true, i.e.,* $\mathcal{I}, \pi \models q_i$ *for some* $1 \leq i \leq n$.

Let $\boldsymbol{x} = x_1, \ldots, x_n$ *be the answer variables of* q, *and let* $\boldsymbol{a} = a_1, \ldots, a_n$ *be a tuple of individuals. Then we write* $\mathcal{I}, \pi \models q(\boldsymbol{a})$ *if* π *is a match for* q *in* $\mathcal{I}$ *and* $\pi(x_i) = a_i$ *for each* $1 \leq i \leq n$, *and we write* $\mathcal{I} \models q(\boldsymbol{a})$ *if there is a match* π *such that* $\mathcal{I}, \pi \models q(\boldsymbol{a})$. *If* q *is Boolean, i.e., if* $n = 0$, *then we write simply* $\mathcal{I}, \pi \models q$ *and* $\mathcal{I} \models q$.

We sometimes represent a CQ q as a graph.

Definition 22 (Query Graph). *The* query graph *of a CQ* q *is the node-arc labeled graph* $\mathsf{G}(q) = \langle V, E, \lambda \rangle$ *with nodes* $V = \mathsf{Vl}(q)$ *and edges* $E = \{(v, v') \mid R(v, v') \in q \text{ for some role } R\}$, *each node* v *is labeled with the set of concepts* $\lambda(v) = \{A \mid A(v) \in q\}$ *and each arc* (v, v') *is labeled with the set of roles* $\lambda(v, v') = \{R \mid R(v, v') \in q\}$.

Example 18. The query graphs of q_3^{theo} and q_6^{theo} (see Example 17) are depicted in Figure 8.

Note that finding a match for q in an interpretation $\mathcal{I}$ reduces to finding a homomorphic mapping of $\mathsf{G}(q)$ into $\mathcal{I}$, as the one depicted in Figure 9.

5.2 Reasoning with Queries

We next consider some reasoning problems associated to queries over DL knowledge bases. First, for queries with answer variables, the basic decision problem is

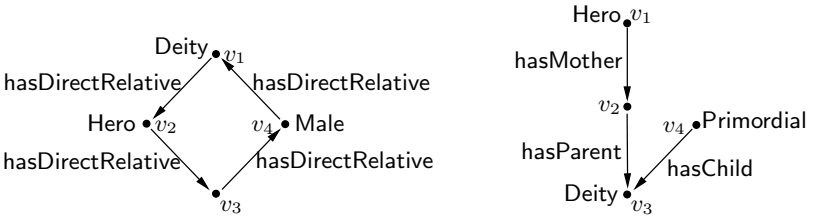

Fig. 8. The query graphs of q_3^{theo} and q_6^{theo}

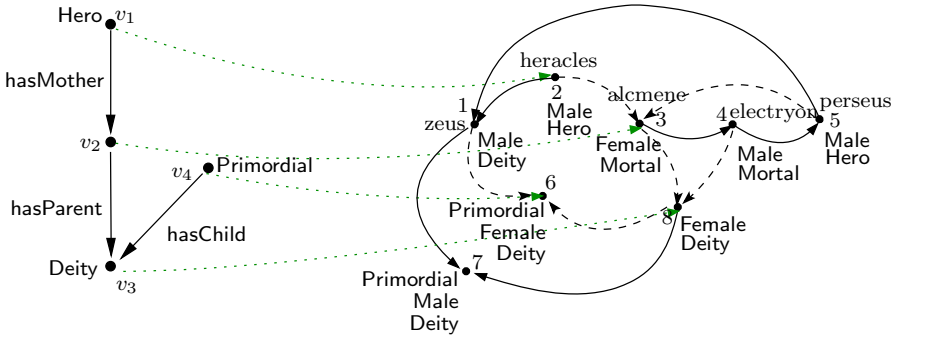

Fig. 9. An example of a query match for q_6^{theo} in $\mathcal{I}_1$

usually called *query answering*, and it consists of deciding whether a given tuple is an answer to the query in *all* the models of the knowledge base, sometimes referred to as a *certain answer*. Boolean queries (which have no answer variables) evaluate to true or to false in an interpretation. For them we consider the query entailment problem, where we want to decide whether the query evaluates to true in *all* the models of a knowledge base.

Query Answering and Query Entailment. For non-Boolean queries, the basic reasoning task is query answering:

Definition 23 (Query Answering). *Given a query q with answer variables $\boldsymbol{x} = x_1, \ldots, x_n$ and a knowledge base $\mathcal{K}$, we say that $\boldsymbol{a} = a_1, \ldots, a_n$ is* an answer for q in $\mathcal{K}$*, in symbols $\mathcal{K} \models q(\boldsymbol{a})$, if $\mathcal{I} \models q(\boldsymbol{a})$ for every model $\mathcal{I}$ of $\mathcal{K}$. By* $\mathsf{ans}(q, \mathcal{K})$ *we denote the set of all answers for q in $\mathcal{K}$.*

The query answer enumeration problem *for $\mathcal{K}$ and q consists of listing all the answers for q in $\mathcal{K}$. The associated decision problem, which we call simply* query answering*, consists of deciding, given a query q, a knowledge base $\mathcal{K}$, and a tuple $\boldsymbol{a}$, whether $\boldsymbol{a}$ is an answer for q in $\mathcal{K}$.*

For Boolean queries, whose only possible answer is the empty tuple, one usually talks about *query entailment*.

Definition 24 (Query Entailment). *Given a Boolean query q and a knowledge base $\mathcal{K}$, we say that $\mathcal{K}$* entails *q, in symbols $\mathcal{K} \models q$, if $\mathcal{I} \models q$ for every model $\mathcal{I}$ of $\mathcal{K}$.*

The query entailment *problem consists of deciding, given a knowledge base $\mathcal{K}$ and a Boolean query q, whether $\mathcal{K} \models q$.*

We often call an interpretation $\mathcal{I}$ such that $\mathcal{I} \models \mathcal{K}$ but $\mathcal{I} \not\models q$ a *countermodel* (to the entailment of q in $\mathcal{K}$).

Example 19. Recall the interpretation $\mathcal{I}_1$ from Example 6 (see Figure 3), and the query q_4^{theo} from the previous example. The binding π with $\pi(x_1) = \text{zeus}^{\mathcal{I}} = 1$, $\pi(x_2) = \text{alcmene}^{\mathcal{I}} = 3$ and $\pi(v) = \text{heracles}^{\mathcal{I}} = 2$ is a match for q_4^{theo} in $\mathcal{I}_1$. In fact, we can set $\pi(v_1) = \text{zeus}^{\mathcal{I}}$, $\pi(v_2) = \text{alcmene}^{\mathcal{I}}$ and $\pi(v_3) = \text{heracles}^{\mathcal{I}}$ to obtain a match in *any* model of $\mathcal{K}_2^{theo}$. Hence the pair $(\text{zeus}, \text{alcmene})$ is an answer of q_4^{theo}, in symbols $\mathcal{K}_2^{theo} \models q_4^{theo}(\text{zeus}, \text{alcmene})$.

It is well known that the query answering problem for an arbitrary query reduces linearly to the entailment problem of a Boolean query.

Proposition 1. *Let $\exists \boldsymbol{v}.\varphi(\boldsymbol{x}, \boldsymbol{v})$ be a query with answer variables $\boldsymbol{x} = x_1, \ldots, x_n$, and let $\boldsymbol{a} = a_1, \ldots, a_n$ be a tuple of individuals of the same arity. Then, for each knowledge base $\mathcal{K}$, $\mathcal{K} \models q(\boldsymbol{a})$ if and only if $\mathcal{K} \models q^{\boldsymbol{a}}$, where $q^{\boldsymbol{a}}$ is the Boolean query $\exists \boldsymbol{v}.\varphi'(\boldsymbol{v})$ obtained by replacing in φ each occurrence of x_i by a_i, for each $x_i \in \boldsymbol{x}$.*

Hence, in order to characterize the computational complexity of query answering, it suffices to consider the entailment of Boolean queries.

Note that deciding query entailment is always at least as hard as deciding knowledge base satisfiability, as the latter can be reduced to the former:

Proposition 2. *Let $\mathcal{L}$ be a DL, and let $\mathcal{K}$ be an $\mathcal{L}$ knowledge base. Then $\mathcal{K}$ is satisfiable if and only if $\mathcal{K} \not\models \exists v.\bot(v)$.*

5.3 Combined and Data Complexity

The computational complexity of a problem is always measured in terms of the *size* of the problem instance. However, when reasoning with queries or in the presence of extensional data, the definition of the input that is relevant for evaluating the complexity may differ from one scenario to another. In the database setting, when queries in some query language are to be answered over a family of databases, Vardi identified two main settings where it is meaningful to measure the complexity in different ways [97]:

- If one is interested in the complexity of evaluating any given query in the query language over any arbitrary database, then both components are part of the input, and complexity should be measured in the terms of the combined size of the query and the database. This is known as the *combined complexity* of query answering.

– If one is interested in the complexity of evaluating some fixed query over any input database, then only the size of the data determines the size of the input, and complexity should be measured in the terms of the size of the database only. This is known as the *data complexity* of query answering.

In the setting of querying DL knowledge bases, these are also accepted as significant measures of complexity. Combined complexity is considered the classical way to measure complexity, and it takes as an input a full knowledge base and a query. Data complexity considers only the extensional data in the ABox as an input, and both the TBox (intensional axioms) and the query are assumed to be fixed. Data complexity becomes relevant when we see a knowledge base as a data repository, because then the extensional data can become very large, and it is usually much larger than the query and than the intensional axioms expressing constraints on the data. Therefore, the contribution of the extensional assertions to the complexity of inference should be singled out, and one must pay attention to optimizing inference techniques with respect to data size, as opposed to the overall size of the knowledge base.

The measures of complexity that we consider are the following:

Definition 25 (Combined complexity of query entailment). *The* (combined) complexity *of query entailment is the complexity (in terms of complexity classes) of deciding, given a knowledge base $\mathcal{K}$ and a query q, whether $\mathcal{K} \models q$.*

For *data complexity*, we assume that $\mathcal{T}$ and q are fixed, and measure the complexity of deciding $\langle \mathcal{A}, \mathcal{T} \rangle \models q$ in terms of the size of $\mathcal{A}$. In order for this characterization to be meaningful, we must ensure that all the intensional information is indeed given in $\mathcal{T}$ and that $\mathcal{A}$ only contains simple assertions.

Definition 26 (Data complexity of query entailment). *The* data complexity *(in terms of complexity classes) of query entailment is the complexity of deciding $\langle \mathcal{A}, \mathcal{T} \rangle \models q$, where $\mathcal{T}$ and q are assumed to be fixed, and the input $\mathcal{A}$ is an* extensionally reduced ABox *where all assertions are of the form $A(a)$ or $p(a, b)$ for A a concept name and p a role name.*

6 The Challenges of Query Answering

Most DLs are fragments of FOL and a KB $\mathcal{K}$ can be rewritten as a FO formula $\varphi_{\mathcal{K}}$ in a straightforward way [7]. CQs and UCQs are also special classes of FO formulas. Hence the CQ entailment problem is a special case of the FOL logical consequence problem: we are given a KB $\mathcal{K}$ and a FOL formula q (the query), and we want to decide whether $\varphi_{\mathcal{K}}$ implies q, in the standard FOL semantics. This can be naturally reduced to other reasoning problems like validity or unsatisfiability of a FOL formula. However, all of these problems are undecidable for full FOL and it is thus important that $\varphi_{\mathcal{K}}$ and q fall into a decidable fragment. The decidability of many DLs hinges on the fact that the formula $\varphi_{\mathcal{K}}$ resulting from $\mathcal{K}$ falls into the so-called *guarded fragment* of FOL [43]. Since CQs in general do

not satisfy the guardedness condition, the decidability of the fragment that is required to accommodate $\varphi_{\mathcal{K}}$ and q is not apparent. It does not seem possible to use existing off-the-shelf algorithms and tools for decidable fragments of FOL, and it is necessary to develop special techniques that allow us to overcome the difficulties arising from the DL, from the query, and from their interaction.

We can identify two main reasons that make query answering a challenging problem:

6.1 Multiple Models

Query answering algorithms from databases usually evaluate the query over one single input structure, the database. In the presence of DL ontologies, however, the query has to be evaluated over all the structures that are models of the data and the ontology. A knowledge base $\mathcal{K}$ has, in principle, infinitely many models. A trivial reason for this is that we can use arbitrary sets as domains in interpretations, and we can add additional, possibly irrelevant information to any model in a way that it is still a model. But even if we disregard these trivial reasons and consider only the 'meaningful' differences between models, resulting from different ways in which the data may be completed to satisfy all constraints given by the ontology, we can still end up with a very large number of models.

To develop query answering techniques it is usually possible to show that a finite subset of all the models suffices. Researchers do this sometimes explicitly, sometimes 'implicitly', by showing that if there is a countermodel showing non-entailment of a query, then there is also a countermodel with a regular and easy-to-manage structure. In fact, most query answering techniques, specially for expressive DLs, focus on building a countermodel.

In the case of Horn DLs, and in particular for the lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families that we discuss below, we can go even further and show the following key property:

> *Canonical model property:* given a satisfiable knowledge base $\mathcal{K}$, one can construct a *canonical model* $\mathcal{I}_{\mathcal{K}}$ such that $\mathsf{ans}(q, \mathcal{I}_{\mathcal{K}}) = \mathsf{ans}(q, \mathcal{K})$ for every positive FOL query q.

That is, for every CQ q, we have that q is entailed by $\mathcal{K}$ if and only if q has a match in the canonical model $\mathcal{I}_{\mathcal{K}}$, hence $\mathcal{I}_{\mathcal{K}}$ suffices for answering all queries. This property plays a central role in the data complexity of query answering. In fact, all tractability results that we mention in Section 7 are for logics that have the canonical model property.

In contrast, the presence of any kind of disjunctive information in the ontology results in the existence of models that differ in the queries they entail, and makes the query entailment problem coNP hard in data complexity, even for the simplest instance queries. As we will see, this applies to all the expressive logics we discuss in Section 8. We will also see that the need to consider several models has a big impact on the combined complexity of CQ answering and makes the problem exponentially harder than standard reasoning for most expressive DLs.

6.2 Large and Infinite Models

Unfortunately, identifying one model, or one set of models, that is sufficient for deciding query entailment, is not enough to make query answering easy. Usually we must overcome an additional difficulty: (the domain of) models can be, in general, infinite. This is caused by existential concepts in the right hand side of GCIs, a feature that is considered fundamental in DLs and allowed even in the lightweight logics we have mentioned.

Many DLs enjoy the *finite model property*. That is, every satisfiable KB has a model where the domain is a finite set. For lightweight DLs like *DL-Lite* and $\mathcal{EL}$, there is even a model where the domain's cardinality is polynomially bounded in the size of the KB. However, these positive properties are of limited use for query answering. Small and finite models are usually built by 'reusing' domain elements to satisfy existential restrictions. This creates cycles in models that may not affect KB satisfiability, but often result in spurious query matches that are not really implied by the KB. For example, one can avoid generating an infinite R-chain of objects for satisfying an axiom $A \sqsubseteq \exists R.A$ by simply 'reusing' one element e and making it an R-successor of itself. However, this would cause the query $\exists x.R(x,x)$ to be erroneously entailed.

Researchers have come up with a number of different techniques to overcome the challenges above. We will discuss some of them in the next sections. The results obtained so far suggest that the size of models is easier to tame than their number. For DLs that enjoy the canonical model property, even if canonical models are infinite, several query answering algorithms have been proposed and implemented. Intuitively, algorithms can build on the fact that canonical models, even if they are infinite, enjoy a relatively regular structure and can be compactly represented. In contrast, for expressive DLs like $\mathcal{ALC}$ and its extensions, the need to consider multiple models for query answering, and the interaction of these models with the query, results in much more complex algorithms that have eluded implementation until now.

Restricting the Problem. Some existing DL reasoners for expressive DLs, like KAON2[3], Racer[4] and Pellet[5], support some form of conjunctive query answering for DLs that are not Horn. However, none of them supports them fully. Instead, they impose some restrictions that make the problem easier, such as:

- The semantics is modified, and bindings for query matches are required to map all variables to the interpretation of some ABox individual. In this way, the possible infinite structures generated via existential quantification become irrelevant, and query answering can be done over a set of small structures.
- The proper semantics of query matches is preserved, but only queries that satisfy certain syntactic restrictions are allowed. For example, if only queries

[3] `http://kaon2.semanticweb.org`
[4] `http://www.racer-systems.com/`
[5] `http://clarkparsia.com/pellet`

whose graph is tree-shaped are allowed, query answering can be polynomially reduced to standard DL reasoning tasks.

These restricted settings are out of the scope of this chapter. Here we focus on query answering techniques for arbitrary CQs with standard FOL semantics. In the next sections, we give a survey of such techniques and complexity results, for lightweight and expressive DLs.

7 Lightweight and Horn DLs

In certain scenarios, the need for efficient and scalable reasoning dominates over the need for expressive formalisms, and the prohibitive computational complexity of expressive Description Logics makes them of limited use when very large data repositories are accessed through DL ontologies. This has been the main motivation for identifying DLs that have sufficient expressive power to express rich data models, but that have better computational properties. In particular, data complexity must be kept tractable and as low as possible. In this section we briefly present three well known families of DLs that allow for tractable query answering in data complexity. The first two, *DL-Lite* and $\mathcal{EL}$, are closely related to the OWL 2 profiles QL and EL, respectively, and are discussed in more depth in the tutorial *OWL 2 Profiles: An Introduction to Lightweight Ontology Languages* of this summer school [58].

7.1 Query Answering in *DL-Lite*

The DLs of the *DL-Lite* family are considered the most important formalisms for data access though DL ontologies [13]. They were specially tailored in such a way that they can describe complex conceptual models, yet query answering can be achieved by efficient and scalable algorithms. Indeed, many central features of conceptual modeling formalisms used in databases and software engineering, like entity-relationship and UML class diagrams, can be expressed in *DL-Lite*. These features include, for example, ISA-relations between classes, class disjointness, domain and range restrictions on roles, functionality constraints, and mandatory (non-)participation of the objects in some class in some role.

The basic *DL-Lite*, usually denoted *DL-Lite*$_{core}$, is the fragment of $\mathcal{ALC}$ that supports only GCIs

$$B \sqsubseteq C$$

where B is a concept of one of the follow the forms

$$A \qquad \exists p \text{ (or equivalently, } \exists p.\top) \qquad \exists p^- \text{ (or equivalently, } \exists p^-.\top)$$

for some concept name $A \in \mathsf{N_C}$ and some role name $p \in \mathsf{N_R}$, while C is a *possibly negated* concept of the same form as B, i.e., C is of one of the forms:

$$A \qquad \exists p \qquad \exists p^- \qquad \neg A \qquad \neg\exists p \qquad \neg\exists p^-$$

Thanks to the limited expressiveness of this logic, reasoning is relatively easy. Traditional reasoning tasks like KB satisfiability and instance checking are feasible in polynomial time in both data and combined complexity. As we discuss below, query answering is also tractable and even feasible in logarithmic space.

There are many extensions of this core DL that preserve its good properties. The best known ones are *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$. *DL-Lite*$_{\mathcal{R}}$ extends *DL-Lite*$_{core}$ by allowing RIAs of the form

$$P \sqsubseteq P' \qquad\qquad P \sqsubseteq \neg P'$$

where P is a role name $p \in \mathsf{N_R}$ or its inverse p^-. *DL-Lite*$_{\mathcal{F}}$ extends *DL-Lite*$_{core}$ by allowing a new kind of axioms

$$(\mathsf{funct}\ P)$$

where, as above, P is a role name $p \in \mathsf{N_R}$ or its inverse p^-. This axiom is just a shortcut for the GCI $\top \sqsubseteq \leqslant 1\, P.\top$ stating that no element can be related via P to more than one element, i.e., the role P is *functional*. Other extensions that preserve the low computational complexity of *DL-Lite*$_{core}$ allow, for example, for both RIAs and functionality assertions but restrict their interaction, or allow for negative assertions in the ABox. In other extensions of *DL-Lite* some of the good computational properties are preserved, but others are lost. For example, if concept of the form $A \sqcap B$ are allowed in axioms, reasoning remains tractable, but it is not feasible in logarithmic space anymore. We refer to [13,4] for a detailed discussion of many extensions of *DL-Lite* and their computational complexity. In what follows, when we talk about *DL-Lite*, we mean *DL-Lite*$_{core}$ and its basic extensions *DL-Lite*$_{\mathcal{R}}$ and *DL-Lite*$_{\mathcal{F}}$.

DL-Lite enjoys a remarkable property: we can compile away the intensional knowledge in a given ontology $\mathcal{T}$ that is relevant for answering a given query q, so that the query can be valuated over the data only.

> *FOL Rewritability:* Given a *DL-Lite* ontology $\mathcal{T}$ and a CQ q, one can compute a FOL query $q^{\mathcal{T}}$ such that, for every ABox $\mathcal{A}$, $\mathsf{ans}(q, \langle \mathcal{A}, \mathcal{T} \rangle) = \mathsf{ans}(q^{\mathcal{T}}, \langle \mathcal{A}, \emptyset \rangle)$.

This means that we can reduce the problem of answering a query over a *DL-Lite* knowledge base to simply evaluating a FOL query, or equivalently, an SQL query, over a standard database, a well understood problem for which efficient solutions are readily available. Since the rewriting of q into $q^{\mathcal{T}}$ does not depend on the ABox $\mathcal{A}$, this also implies that the data complexity of query answering over *DL-Lite* KBs is not higher than the data complexity of evaluating FOL queries over standard databases, which is a problem with very low complexity (in the complexity class called AC_0 [54], see e.g., [100] for a definition).

FOL Rewritability has been exploited to propose and implement query answering systems. This was first done in [13], where Calvanese et al. proposed the *perfect reformulation* (PerfectRef) algorithm for reformulating a CQ q into a UCQ $q^{\mathcal{T}}$. Intuitively, the algorithm uses $\mathcal{T}$ to replace concepts and roles in

the query by concepts and roles that imply them. Different choices for replacing a given concept or role result in different queries. To obtain a complete query that covers all possible ways of implying the concept and roles in the input q, it is necessary to allow the unification of query variables in different atoms. We illustrate this algorithm with an example, and refer to [58] for a more detailed presentation and proofs of correctness.

$\mathsf{Deity}\,(\text{zeus})$	
$\mathsf{Deity}\,(\text{rhea})$	$\mathsf{Primordial} \sqsubseteq \mathsf{Deity}$
$\mathsf{Primordial}\,(\text{gaia})$	$\exists\mathsf{hasParent} \sqsubseteq \neg\mathsf{Primordial}$
$\mathsf{hasMother}\,(\text{zeus}, \text{rhea})$	$\mathsf{hasMother} \sqsubseteq \mathsf{hasParent}$
$\mathsf{hasMother}\,(\text{chronos}, \text{gaia})$	$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}$
(a) ABox	(b) TBox

Fig. 10. Knowledge Base for Example 20

Example 20. We consider the ABox and TBox in Figure 10, which is a version of Example 13 simplified to the *DL-Lite* syntax. Recall the following query from Example 13:

$$q_1^{theo}(x) : -\exists v.\mathsf{hasAncestor}(x, v) \wedge \mathsf{Deity}(v).$$

Intuitively, the PerfectRef algorithm applies the axioms 'backwards' on the query, to obtain new subqueries that can contribute to the query answers. For example, we can apply $\mathsf{Primordial} \sqsubseteq \mathsf{Deity}$ to the last query atom and obtain a new query

$$q'(x) : -\exists v.\mathsf{hasAncestor}(x, v) \wedge \mathsf{Primordial}(v).$$

By applying consecutively $\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}$ and $\mathsf{hasMother} \sqsubseteq \mathsf{hasParent}$ to the first atom of q', we can derive

$$q''(x) : -\exists v.\mathsf{hasMother}(x, v) \wedge \mathsf{Primordial}(v).$$

Applying in this way to each atom of each query all the applicable axioms, we can obtain a UCQ that retrieves all query answers when posed over the ABox data only. For our example, the resulting UCQ contains the following six CQs:

$$\begin{aligned}
q_1(x) &: -\ \exists v.\mathsf{hasAncestor}(x, v) \wedge \mathsf{Deity}(v)\\
q_2(x) &: -\ \exists v.\mathsf{hasAncestor}(x, v) \wedge \mathsf{Primordial}(v)\\
q_3(x) &: -\ \exists v.\mathsf{hasParent}(x, v) \wedge \mathsf{Deity}(v)\\
q_4(x) &: -\ \exists v.\mathsf{hasParent}(x, v) \wedge \mathsf{Primordial}(v)\\
q_5(x) &: -\ \exists v.\mathsf{hasMother}(x, v) \wedge \mathsf{Deity}(v)\\
q_6(x) &: -\ \exists v.\mathsf{hasMother}(x, v) \wedge \mathsf{Primordial}(v)
\end{aligned}$$

Chronos is in the answer to q_6 and Zeus in the answer to q_5, hence both are in the answer to the query.

The PerfectRef algorithm has been implemented in the QuOnto system, that now supports several optimizations of the original technique. Even though the low data complexity suggests that query evaluation for *DL-Lite* can be done very efficiently, things are not so straightforward in practice. The PerfectRef algorithm produces a UCQ whose size is exponential in the query, and often this results in huge UCQs that cannot be efficiently evaluated using existing database technologies. Hence a good amount of recent research has been devoted to understanding the reasons for such large queries, improving the PerfectRef algorithm, and developing alternative rewriting algorithms whose implementations exhibit better performance. For example, in [25], Chortaras, Trivela and Stamou proposed optimizations that reduce the number of obtained queries, mainly due to the unification step, without compromising the completeness of the approach; these optimization have been implemented in the Rapid system. Perez-Urbina, Motik and Horrocks [81] use a different technique, based on resolution, to do the reformulation step. It also works for more expressive DLs than *DL-Lite*, which are not FOL rewritable, but for the latter it produces a recursive Datalog query rather than a FOL query. This technique is implemented in the Requiem system. Another optimized version of PerfectRef, but extended to the related language Datalog$^{\pm}$, is implemented in the Nyaya system [40]; see also [41]. Very promising results have been obtained by reformulating q not as a UCQ, but as a query in a more expressive fragment of FOL. Here we find the work of Rosati and Almatelli [88], that uses a non-recursive Datalog query as target language. It has been implemented in a system called Presto. Although still exponential in the worst case, experiments reveal that the system can be efficient in more cases. On the theoretical side, Gottlob and Schwentick [42] have developed a technique to obtain a polynomial size rewriting for expressive variants of *DL-Lite*, but it uses additional constants and symbols, and does not seem suited for implementation. Kikot, Kontchakov, and Zakharyaschev showed recently that for *DL-Lite*$_{core}$ there is always a polynomial rewriting that does not need any additional symbols, but for *DL-Lite*$_{\mathcal{R}}$ this does not hold and exponential rewritings are unavoidable [56].

7.2 Query Answering in $\mathcal{EL}$

$\mathcal{EL}$ and its extensions are the other prominent family of lightweight DLs. Unlike *DL-Lite*, $\mathcal{EL}$ was not tailored explicitly for data access. Instead, it was proposed as a DL that could express many of the modeling features that are required in life science ontologies, and that still supports efficient reasoning [6].

Essentially, $\mathcal{EL}$ is half of $\mathcal{ALC}$: it supports conjunction, but no disjunction, and existential quantification, but no universals. It also disallows full negation, as having it would enable us to express both disjunctions and universal quantification making it as expressive (and computationally hard) as $\mathcal{ALC}$. However, in most extensions of the basic $\mathcal{EL}$, the special concept name $\bot$ can be used to simulate a restricted form of negation. When the $\bot$ concept is allowed, the combined complexity of standard reasoning and the data complexity of both standard reasoning and query answering, are all complete for polynomial time.

If $\bot$ is not allowed, reasoning problems like KB satisfiability become trivial, as there are no unsatisfiable KBs.

Prominent extensions of $\mathcal{EL}$ include $\mathcal{ELH}$, which supports role hierarchies, $\mathcal{EL}^+$ allows role inclusions of the form $p_1 \circ \cdots \circ p_n \sqsubseteq r$, similarly as in $\mathcal{SROIQ}$ but without the regularity restriction, and $\mathcal{EL}^{++}$, which additionally supports nominals [6] and is the basis of the OWL 2 EL profile (see [58]); some of these extensions support also other features not discussed in this chapter. The tractability of traditional reasoning problems is preserved in all these DLs. In contrast, other very natural extensions of the basic $\mathcal{EL}$, like the inverse roles present in $\mathcal{ELI}$ and $\mathcal{ELHI}$, preserve tractability of data complexity but make it intractable in combined complexity [6,53].

The complexity of CQ answering in $\mathcal{EL}$ and its extensions is relatively well understood [87,57,59]. For the basic $\mathcal{EL}$ and for $\mathcal{ELH}$, it is complete for polynomial time in data complexity and for NP in combined complexity [87]. For $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$, however, the problem becomes undecidable [87,57,59]. Decidability can be regained by imposing some regularity conditions in the RIAs, similarly as in $\mathcal{SHOIQ}$ [59]. A detailed analysis of the data complexity of CQs in many different extensions of $\mathcal{EL}$ can be found in [57].

Implemented systems supporting query answering in $\mathcal{EL}$ are not as widespread as for *DL-Lite*. Since query answering in $\mathcal{EL}$ is hard for polynomial time, it is not possible to use query rewriting to compile away the TBox and obtain a FOL query. However, successful approaches do apply some form of query rewriting and solve the problem exploiting existing database technologies. The algorithm mentioned above due to Pérez-Urbina, Motik and Horrocks [81] rewrites the query into a Datalog program, and uses Datalog engines to answer queries over $\mathcal{ELH}$ and $\mathcal{ELHI}$ KBs. Another interesting approach is the one usually called the *combined approach*, introduced in [63], that uses *data completion* to compile the TBox into the *data* rather than into the query. After some polynomial rewritings, the query can be posed over a completed ABox that is polynomially larger than the original one, using standard relational database technologies.

7.3 Query Answering in other Horn DLs

There are DLs more expressive than *DL-Lite* and $\mathcal{EL}$ that are still tractable in data complexity. These are the so called *Horn DLs*, which are obtained by taking an expressive DL and disallowing disjunction, i.e., intersecting it with the Horn fragment of first order logic [60,53]. This family of DLs includes, for example, Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$, Horn-$\mathcal{SRIQ}$ and Horn-$\mathcal{SROIQ}$, obtained from the corresponding expressive logics introduced in Section 2.

These four DLs extend the basic DLs of the *DL-Lite* family, as well as $\mathcal{ELHI}$. Like $\mathcal{EL}$ and *DL-Lite*, they enjoy the *canonical model property* that allows us to do query answering using one single model. An important difference, however, is that building a representation of the canonical model is more expensive. For the DLs of the *DL-Lite* and $\mathcal{EL}$ families, a finite representation of canonical model can be computed in polynomial time. For Horn-$\mathcal{SHIQ}$ and its extensions, computing such a representation may require exponential time and space.

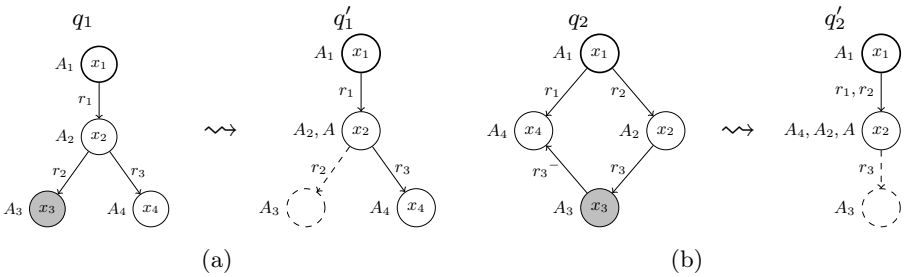

Fig. 11. Query rewriting for Horn-$\mathcal{SHIQ}$

In the mentioned Horn DLs, both standard reasoning and and query answering are P-complete in data complexity. The combined complexity is ExpTime-complete for Horn-$\mathcal{SHIQ}$ and Horn-$\mathcal{SHOIQ}$, and 2ExpTime-complete for Horn-$\mathcal{SRIQ}$ and Horn-$\mathcal{SROIQ}$. The data complexity of standard reasoning in Horn-$\mathcal{SHIQ}$ was established in [53], and its combined complexity in [60]. For CQ answering, both kinds of complexity were characterized in [29]. For Horn-$\mathcal{SHOIQ}$, Horn-$\mathcal{SRIQ}$ and Horn-$\mathcal{SROIQ}$, the complexity of standard reasoning was established in [77], and of CQ answering in [78].

To our knowledge, Horn-$\mathcal{SHIQ}$ is the most expressive DL for which query answering has been implemented. In [33], Eiter et al. proposed a query rewriting technique that transforms a given Horn-$\mathcal{SHIQ}$ ontology $\mathcal{T}$ and a CQ q into a Datalog program ready for evaluation over any ABox, implemented in the system Clipper.[6] The central idea of the rewriting is to clip off variables from x that can be mapped to elements implied by the existentials in the TBox, and to add to the query concept atoms that imply the existence of these elements. In this way, the original query has a match whenever one of the rewritten queries does.

Example 21. The query $q_1(x_1) \leftarrow A_1(x_1)$, $r_1(x_1, x_2)$, $A_2(x_2)$, $r_2(x_2, x_3)$, $A_3(x_3)$, $r_3(x_2, x_4)$, $A_4(x_4)$ is depicted on the left hand side of Figure 11a. If the ontology $\mathcal{T}$ implies $A \sqsubseteq \exists r_2.(B \sqcap A_3)$, then we can add $A(x_2)$ to the query and clip off the variable x_3 (grey node), since the existence of an element that satisfies A already implies the existence of a mapping for x_3.

Similarly for the query $q_2(x_1) \leftarrow A_1(x_1)$, $r_2(x_1, x_2)$, $A_2(x_2)$, $r_3(x_2, x_3)$, $A_3(x_3)$, $r_1(x_1, x_4)$, $A_4(x_4)$, ${r_3}^-(x_3, x_4)$ in Figure 11b. If $\mathcal{T}$ implies $A \sqsubseteq \exists r_3.(B \sqcap A_3)$, then we can clip off x_3 and require x_2 and x_4 to be mapped together to an element satisfying A, since that would suffice to ensure the existence of a match for x_3.

7.4 The Complexity Landscape for Lightweight DLs

We summarize in Table 2 the complexity results we have discussed in this section. Note that instance checking and query answering are tractable in data

[6] `http://www.kr.tuwien.ac.at/research/systems/clipper/`

complexity for all the listed DLs, but only *DL-Lite* enjoys FOL rewritability. The combined complexity of CQs is NP-hard for all DLs, since in fact this already holds in the absence of an ontology, for deciding the entailment of a CQ for a given ABox or database. In Horn DLs that are at least ExpTime-hard, this NP-hardness is subsumed by the complexity of standard reasoning, and CQ entailment is not harder than KB satisfiability. For comparison, we also include the complexity of evaluating CQs over plain databases (the instance checking column is empty since the problem is not defined).

Table 2. The Complexity of Query Entailment in Lightweight DLs

	Instance queries		**CQs**	
	Combined complexity	Data complexity	Combined complexity	Data complexity
Plain DBs			NP-complete	in AC_0
DL-Lite	in P	in AC_0	NP-complete	in AC_0
$\mathcal{EL}$	P-complete	P-complete	NP-complete	P-complete
$\mathcal{EL}^+, \mathcal{EL}^{++}$	P-complete	P-complete	undecidable	
regular-$\mathcal{EL}^+$, regular-$\mathcal{EL}^{++}$	P-complete	P-complete	PSpace-complete	P-complete
Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$	ExpTime-complete	P-complete	ExpTime-complete	P-complete
Horn-$\mathcal{SRIQ}$, Horn-$\mathcal{SROIQ}$	2ExpTime-complete	P-complete	2ExpTime-complete	P-complete

8 Query answering in Expressive DLs

Finally, we discuss query answering in *expressive DLs* that are at least as expressive as $\mathcal{ALC}$. These logics have the following features:

- They do not enjoy the canonical model property.
- Instance checking is coNP hard in data complexity [91], and so is CQ entailment.
- For most of them, the combined complexity of query answering is exponentially higher than the complexity of standard reasoning. The only known exception is $\mathcal{ALCHQ}$ and its sublogics. For some logics no tight complexity have been obtained.
- There are no systems implementing full query answering.

8.1 Forest Shaped Countermodels

Since there is no canonical model, entailment of the query would have to be tested over many models. The strategy of most algorithms is not to do this directly, but to search for a *countermodel* instead. That is, a model of the given KB where the query has no match.

All algorithms developed so far build on the fact that one can restrict the search for countermodels to interpretations that have a regular, forest like structure, which we call *forest interpretations*. The domain $\Delta^{\mathcal{I}}$ of such a model is a *forest*, that is, a finite set of possibly infinite trees. The branching of the trees is bounded by some number that depends polynomially on the input KB $\mathcal{K}$. We require the interpretation of individuals and of role names to satisfy the following restrictions:

- for each individual a, $a^{\mathcal{I}}$ is the root of some tree.
- a pair w, w' of nodes in the forest can only be in the interpretation $r^{\mathcal{I}}$ of some role name r if one of the following conditions holds:
  1. w is a child of w',
  2. w' is a child of w,
  3. at least one of w and w' is a root of some tree, or
  4. r is a non-simple role (i.e., it is implied by some transitive role) and the pair $(w, w') \in r^{\mathcal{I}}$ is implied by a set of pairs, all of which satisfy one of the conditions above.

This is slightly strengthened or relaxed in different DLs. For example, the last condition is not needed in DLs that do not support transitive roles. In $\mathcal{SRIQ}$ and its extensions we also allow nodes to be connected to themselves, i.e., pairs of the form (w, w) may participate in the interpretation of role names.

However, disregarding these minor issues, the general structure of forest interpretations is the same in all the expressive DLs that we discuss here. It can be seen as comprising two parts:

- The roots of the trees provide the interpretation of the individual names, and they may be arbitrarily interconnected. We call this the *graph part* of the interpretation. It is sometimes referred to as the *ABox part*.
- There is a tree-shaped structure attached to each node in the graph part. In each tree, nodes may only be connected to their parents and children, and there may be additional arcs to nodes in the graph part. Intuitively, all the non-root nodes in the trees are objects whose existence is implied by the existential restrictions in the TBox. These trees together form what is often called the *tree-shaped part*, the *anonymous part*, or the *existential part* of the interpretation.

Example 22. Consider the $\mathcal{SROQ}$ KB $\mathcal{K}_g$ is shown in Figure 12. Figure 13 partially depicts a forest-shaped model $\mathcal{I}_g$ of $\mathcal{K}_g$. Its roots are $1 = \text{gaia}^{\mathcal{I}_g}$, $2 = \text{zeus}^{\mathcal{I}_g}$, $3 = \text{heracles}^{\mathcal{I}_g}$, $4 = \text{alcmene}^{\mathcal{I}_g}$, $5 = \text{electryon}^{\mathcal{I}_g}$, and $6 = \text{perseus}^{\mathcal{I}_g}$. They are depicted as large dots, and each of them is labeled with the name of the individual it interprets as well as with the concept names to whose interpretation it belongs. Other domain elements are represented by smaller dots, and are also labeled with the concept names to whose interpretation they belong. For readability, we use the following label names: $L_1 = \{\mathsf{Male}, \mathsf{Deity}\}$, $L_2 = \{\mathsf{Female}, \mathsf{Deity}\}$, $L_3 = \{\mathsf{Female}, \mathsf{Mortal}\}$, $L_4 = \{\mathsf{Male}\}$, and $L_5 = \{\mathsf{Female}\}$. The interpretation represented here is infinite, but only some of its domain elements are depicted. Every non-root element has two successors, which are the

$$\begin{array}{rr} \mathsf{Male}\,(\mathrm{zeus}) & \mathsf{hasFather}\,(\mathrm{heracles}, \mathrm{zeus}) \\ \mathsf{Deity}\,(\mathrm{zeus}) & \mathsf{hasMother}\,(\mathrm{heracles}, \mathrm{alcmene}) \\ \mathsf{Female}\,(\mathrm{alcmene}) & \mathsf{hasFather}\,(\mathrm{alcmene}, \mathrm{electryon}) \\ \mathsf{Mortal}\,(\mathrm{alcmene}) & \mathsf{hasFather}\,(\mathrm{electryon}, \mathrm{perseus}) \\ \mathsf{Hero}\,(\mathrm{heracles}) & \mathsf{hasFather}\,(\mathrm{perseus}, \mathrm{zeus}) \end{array}$$

$$\begin{array}{rcl} \mathsf{Male} & \equiv & \neg\mathsf{Female} \\ \mathsf{Mortal} & \sqsubseteq & \neg\mathsf{Deity} \\ \mathsf{Primordial} & \sqsubseteq & \mathsf{Deity} \\ \neg\mathsf{Primordial} & \sqsubseteq & \exists\mathsf{hasFather}.\mathsf{Male} \sqcap \exists\mathsf{hasMother}.\mathsf{Female} \\ \mathsf{Mortal} & \sqsubseteq & \leqslant 2\,\mathsf{hasParent}.\top \\ \mathsf{Deity} & \sqsubseteq & \forall\mathsf{hasAncestor}.\mathsf{Deity} \\ \mathsf{Deity} & \sqsubseteq & \exists\mathsf{hasAncestor}.\mathsf{Primordial} \\ \mathsf{Primordial} & \equiv & \{\mathrm{gaia}\} \\ \exists\mathsf{hasParent}.\mathsf{Self} & \sqsubseteq & \bot \\ \exists\mathsf{hasChild}.\mathsf{Self} & \sqsubseteq & \bot \\ \mathsf{hasMother} & \sqsubseteq & \mathsf{hasParent} \\ \mathsf{hasFather} & \sqsubseteq & \mathsf{hasParent} \\ \mathsf{hasAncestor} \circ \mathsf{hasAncestor} & \sqsubseteq & \mathsf{hasAncestor} \end{array}$$

Fig. 12. A $\mathcal{SROQ}$ knowledge base $\mathcal{K}_g$

fulfillers of the hasMother and hasFather relations, respectively. The hasFather relation is represented by solid arrows, the hasMother is represented by dashed arrows, and the hasParent relation is the union of both kinds of arrows. The non-simple role hasAncestor is not explicitely depicted, its contains every pair e, e' such that there is a hasParent path from e to e'.

All query answering techniques for expressive DLs that have been proposed so far rely on some kind of forest models, and show that if a given KB $\mathcal{K}$ does not entail a given query q, then there is a forest model of $\mathcal{K}$ where q has no match. For DLs like $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, this property fails and countermodels cannot be assumed to be forest shaped as above (with finitely many trees). This seems to make query answering harder and, as we discuss below, decidability and complexity bounds remain elusive.

8.2 Query Answering Techniques

Many different techniques have been explored for finding forest countermodels. We discuss some of them below.

Automata Theoretic Techniques. Automata on infinite trees have long been employed to solve satisfiability problems in formalisms closely related to DLs, like program logics, cf. [99,98,90,11]. In DLs, they have been applied to obtain worst-case optimal algorithms for the concept satisfiability problem, i.e., for

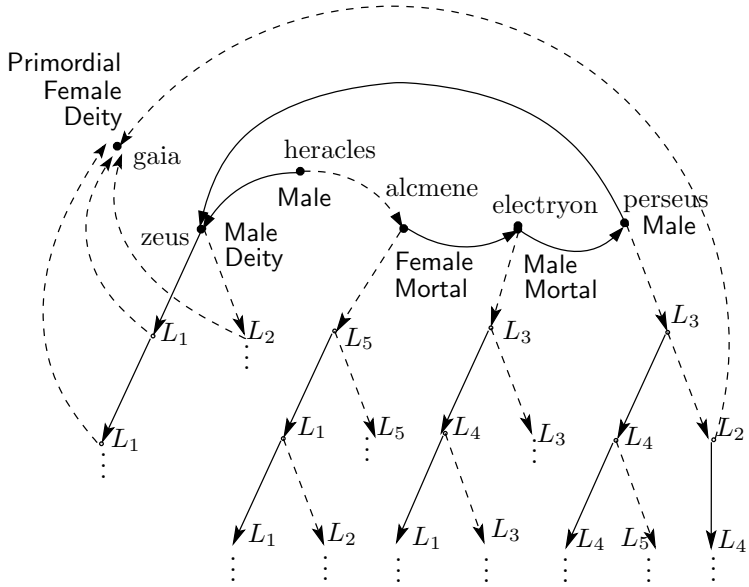

Fig. 13. A forest model $\mathcal{I}_g$ of $\mathcal{K}_g$

reasoning without ABoxes [15,27,28,8]. The basic intuition of these algorithms is as follows: if interpretations can be suitably represented as labeled trees, one can construct from a given a knowledge base $\mathcal{K}$, an automaton that accepts a (possibly infinite) labeled tree if and only if it represents a model of $\mathcal{K}$.

In the last years, these ideas have also been exploited for query answering [22,23,37]. The adaptation is not trivial, since it implies incorporating to the automata ABox reasoning and, more challenging, query matches. The first part, that is, to reason about a full KB with an ABox using tree automata, can be achieved by suitably encoding forest models as trees [22] or, as done more recently [23], by using richer automata models that accept forests rather than just trees [11]. The second part is achieved using techniques similar to the ones proposed in [19], where the authors use automata on infinite words to obtain a tight upper bound for the containment of two P2RPQs over simple semi-structured databases (graphs). Similar ideas have been exploited in other papers to obtain other decidability and complexity results for queries, even with some form of recursion, over equally simple databases [21,20]. Intuitively, one can build an automaton that uses explicit variables in the alphabet to test whether there is a query match in a tree representing an interpretation. We can then project away the explicit variables from the alphabet and intersect the complement of the resulting automaton with the one that tests for modelhood to obtain an automaton that accepts a tree if and only if it represents a countermodel to query entailment. In this way query entailment is reduced to an automaton emptiness test, as done in [22,23].

This approach is quite general, in the sense that it allows to accommodate relatively easily very rich sets of DL constructs and expressive query languages. Such an algorithm was first proposed for a logic closely related to $\mathcal{SRIQ}$, called $\mathcal{ZIQ}$, in [22], and then extended in [23] to full $\mathcal{SRIQ}$, $\mathcal{SROQ}$ and $\mathcal{SROI}$ (and their close relatives $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, $\mathcal{ZOI}$). The authors consider *positive two way regular path queries*, which are a generalization of CQs, UCQS, and of regular path queries, see Section 4.1. The algorithm runs in double exponential time, and hence generalizes the 2ExpTime upper bounds that had been obtained before for weaker logics like $\mathcal{ALCHIQ}$ [52], $\mathcal{ALCHOQ}$ and $\mathcal{ALCHIO}$ [75], $\mathcal{ALC}_{reg}$ [14,16], for $\mathcal{SHIQ}$ [36,22], and for $\mathcal{SHOQ}$ [37]. In [37], Glimm, Horrocks and Sattler use the rolling-up technique that we discuss next, but they combine it with an automata procedure for satisfiability in a DL closely related to $\mathcal{SHOQ}$.

The Rolling-Up Technique. The core idea of the rolling-up technique is to reduce CQ answering to several instances of knowledge base satisfiability. The approach goes back to the work of Calvanese et al. [14,16] where the authors proposed the notion of *tuple graph* of a query as the basis for reducing query containment to unsatisfiability in a variant of Propositional Dynamic Logic. This approach has been called 'rolling-up' since the work of [51], where it was applied to answer CQs in $\mathcal{SH}$, under certain restrictions. In [36] the authors use this technique to obtain a reduction of CQ entailment in $\mathcal{SHIQ}$ to KB satisfiability in the DL $\mathcal{SHIQ}^{\sqcap}$, which extends $\mathcal{SHIQ}$ with role intersection. This yields a decision procedure for 2ExpTime, which was later extended to $\mathcal{SHOQ}$ in [37]. A similar approach was used by Lutz in [67,66] to obtain an ExpTime upper bound for $\mathcal{ALCHQ}$.

Very roughly, the technique works as follows. The image of a query match in a forest shaped interpretation $\mathcal{I}$ can always be seen as a 'forest' consisting of: (a) objects in the graph part of $\mathcal{I}$, and (b) tree-shaped structures in the tree-part of $\mathcal{I}$. Hence, we can proceed in two steps: (1) compute all possible partial mappings of the query into the graph part, and (2) search for trees to expand the graph part into a forest model in such a way that every partial match computed in the first stage does not become a full query match in the extended model. The second part can be achieved by selecting one tree part of each possible match and 'rolling it up' into a concept. A forest shaped countermodel for a given KB $\mathcal{K}$ and query q exists if and only if there is a model of $\mathcal{K}$ where all possible forest-shaped matches fail, or equivalently, if there is a way to extend $\mathcal{K}$ into a satisfiable KB that enforces the negation of a part of each possible match.

Modified Tableau. Another method for deciding query entailment that also appeared in 1998 is the one underlying the Carin algorithms [62]. Carin was proposed as a *hybrid language* combining Description Logics with rules, and its basic reasoning tasks are solved using an algorithm for what the authors call the *existential entailment problem*, which for the purposes of this discussion can be thought of as CQ entailment (it is in fact a simple generalization of containment of CQs in UCQs). The proposed solution to the query entailment problem is

a modified version of an an existing tableau algorithm. Intuitively, a tableau algorithm tries to construct a finite structure that represents a model of a KB, and uses *blocking conditions* to ensure that, if the construction does not fail, then it terminates in finite time and the resulting structure represents a model of the knowledge base. The key idea of Levy and Rousset was that, using the query size as a parameter, the blocking conditions can be modified in such a way the expansion terminates when the structure represents a *set of models that are indistinguishable by the query*, and they showed that a finite representation of a counter-model can be obtained using the modified blocking conditions if the query is not entailed.

The original CARIN algorithm was proposed for a DL called $\mathcal{ALCNR}$, which is closely related to $\mathcal{ALCHQ}$. The technique was first extended to CQs in $\mathcal{SHIQ}$ [74], allowing only simple roles in the query (recall that a role is simple if it is not implied by any transitive role). Then it was extended to the larger class of positive queries (still with only simple roles), and to the DLs $\mathcal{SHIQ}$, $\mathcal{SHOQ}$, and $\mathcal{SHOI}$ [75]. The coNP data complexity upper bound that arises from this algorithms is worst-case optimal, however, no optimal bounds w.r.t. combined complexity can be easily obtained since it builds on tableaux algorithms that are not worst-case optimal. Another drawback of the method is that, since the size of the query is used as a bound on the size of the structures that the query can distinguish, a correct way to handle non-simple roles in the query is not apparent.

Resolution. In [69], a resolution-based algorithm for answering conjunctive queries in $\mathcal{SHIQ}$, but disallowing non-simple roles from the queries, was presented. The general idea is to translate an input KB and the query into a theory of first-order logic and then apply a specially tailored resolution calculus. In this way, query answering can be reformulated in terms of satisfiability in a decidable class of theories of first-order logic. A similar approach was taken in [82], where the authors exploit a resolution calculus together with a datalog engine for query answering in DLs of the *DL-Lite* and $\mathcal{EL}$ families.

Knot Techniques. Knots are a variation of the the *mosaic* technique known from modal logics [71]. Similarly to the rolling-up approach, knot approaches proceed in two steps:

(a) finding partial query matches in the graph part,
(b) finding tree-shaped matches for subqueries in the tree-shaped part.

Knots are mostly useful for (b), and can be combined with other techniques for (a). They were first used for query answering in $\mathcal{ALCH}$ in [79]. This algorithm uses an iterative procedure to compute in a bottom up way the combinations of tree-shaped subqueries of q that are entailed in the tree parts of models, and then computes how these can be extended in the graph part. Transitive roles were incorporated and the technique was extended to $\mathcal{SH}$ in [32].

A different knot based approach was used for $\mathcal{ALCHI}$ in [30], but covering only a special case. Namely, it considers only *simple knowledge bases* with restricted ABoxes where (a) is trivial (as they have only one individual in the ABox), and it focuses on (b). It was generalized to full knowledge bases in [73] using the same ideas that we have described for rolling up [67,66]. This approach can be seen more directly as a search for a forest shaped countermodel, and uses *knot elimination*, which is a variation of the well known *type elimination* technique [84]. It searches for a set of knots that ensures the existence of trees where all possible partial matches of query variables into the graph part of forest shaped models cannot be completed to full query matches. A somewhat intricate refinement of this approach was used in [31] for $\mathcal{S}$, but only for the restricted case of simple KBs.

A positive feature of the above algorithms is that they provide tight complexity bounds and have a positive 'pay-as-you-go' behavior, since they are also worst-case optimal for sublogics of lower complexity. The algorithm for $\mathcal{ALCH}$ in [79] runs in single exponential time; the extension to $\mathcal{SH}$ in [32] runs in 2ExpTime in general, but still in ExpTime if the non-simple roles in the query are suitably bounded. Similarly, the knot elimination approach in [30,73] runs in ExpTime for $\mathcal{ALCH}$ and in 2ExpTime for $\mathcal{ALCHI}$, and is thus worst-case optimal for both. They also give tight coNP upper bounds in combined complexity.

8.3 The Complexity of Query Answering in Expressive DLs

For a few years, only 2ExpTime upper bounds for query answering in expressive DLs were known, and the best matching lower bound was ExpTime-hardness stemming from knowledge base satisfiability.

Inverse Roles. The first conclusive result concerning the precise complexity of the problem was the 2ExpTime lower bound for $\mathcal{ALCI}$ obtained by Lutz [65]. Matching lower bounds were obtained independently in [79] for $\mathcal{ALCH}$, and in [67,66] for $\mathcal{ALCHQ}$.

Very informally, the reason why query answering in $\mathcal{ALCH}$ and $\mathcal{ALCHQ}$ is not harder than standard reasoning is the following. Assume a query variable x has been matched at some node n in the tree part of an interpretation. Then each variable y for which there is an atom $r(x, y)$ in the query has to be matched to a child of n. In the presence of inverses, in contrast, many choices are possible, since each of these neighboring variables could potentially be mapped either at a child or at a parent of n. As a consequence, for $\mathcal{ALCI}$ there are exponentially many ways in which the image of a query match can induce a partial ordering on a set of query variables, while there is only one uniquely determined partial ordering for $\mathcal{ALCHQ}$.

Transitive Roles. The impact of transitivity in the complexity of query answering was a longstanding open problem. From a practical point of view it was acknowledged that it made the design of algorithms harder, and most of the

early query answering algorithms did not allow for non-simple roles in the query [62,93,52,75]. It was shown in [31] that if we allow for of a transitive role and a role inclusion in the TBox, query answering in any extension of $\mathcal{ALC}$ becomes 2ExpTime hard. Similarly as for $\mathcal{ALCI}$, this is roughly because the way query matches in the tree parts order the query variables is not uniquely determined anymore.

As it turns out, transitive roles alone (i.e., without role inclusions) are a source of complexity, but a rather subtle one. CQ entailment in $\mathcal{S}$ is co-NExpTime-hard [31], but there are no matching upper bounds and the tight complexity of the problem remains open. A co-NExpTime upper bound is known only for the case where there is only one role available [9], and an ExpTime bound for the restricted case of simple KBs was obtained in [31].

The Interaction of Inverses and Transitive Roles. The proofs of the 2ExpTime lower bounds for $\mathcal{ALCI}$ in [65] and for $\mathcal{SH}$ in [31] both use an unbounded number of existentially quantified variables in the query, and it is not clear whether these results still hold if a bound on the number of variables is imposed. But things are quite different when both sources of complexity are present: it was shown in [38] that for the logic $\mathcal{SHI}$ query answering is 2ExpTime hard already for queries with only two variables.

The Interaction of Inverses, Nominals, and Number Restrictions. When inverses, nominals and counting features such as functionality or number restrictions are allowed simultaneously, then models may not have a forest shape as we have described. For standard reasoning, it is well known that this has an impact in the computational complexity of reasoning. For example, KB satisfiability is NExpTime complete for $\mathcal{SHOIQ}$ [94,95], while it is only ExpTime complete for its sublogics that disallow one of these three constructors, i.e., $\mathcal{SHIQ}$ [95], $\mathcal{SHOQ}$ [37] and $\mathcal{SHIO}$ [45].

For query answering, the lack of forest models has proven to be a very difficult challenge to overcome. For the DLs $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$, decidability of query answering remains open. Glimm and Rudolph proposed an algorithm in [89] for the sublogic $\mathcal{ACHOIQ}$. It shows decidability, but its does not give an upper bound on its complexity. The best known lower bound is co-N2ExpTime-hardness [39]. The automata algorithm in [23] is defined for $\mathcal{ZOIQ}$, a DL that is slightly more expressive (although less succinct) than $\mathcal{SROIQ}$, but is only complete for KBs that have the forest model property, and in particular for the sublogics $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, and $\mathcal{ZOI}$. For the full $\mathcal{ZOIQ}$ query entailment is in fact undecidable [76], and the decidability of standard reasoning problems remains open.

8.4 The Complexity Landscape for Expressive DLs

The complexity results that we have discussed in this section are summarized in Figure 14. The solid arrows indicate the sublogic relation, while the thick dashed arrows indicate that the $\mathcal{Z}$ logics are at least as expressive as their $\mathcal{SR}$

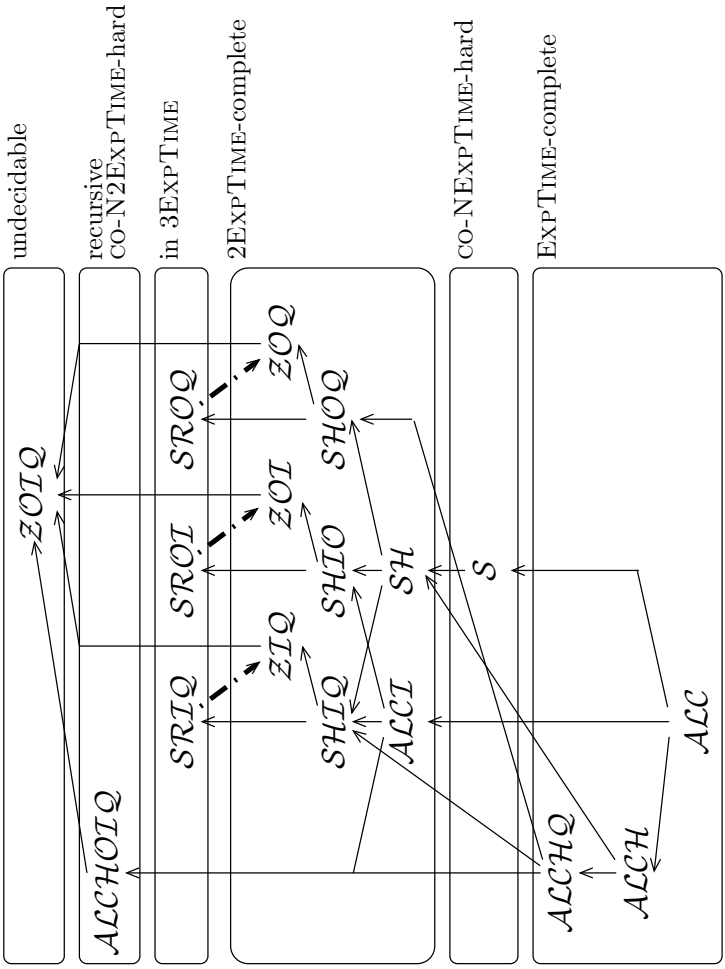

Fig. 14. The complexity of query answering in expressive DLs

Table 3. The Complexity of Query Entailment in Description Logics
(l.b.=lower bound, u.b.=upper bound)

	Combined complexity		**Data complexity**	
	instance queries / standard reasoning	conjunctive queries	instance queries / standard reasoning	conjunctive queries
Plain databases		NP-complete [2]		in AC_0 [54]
DL-Lite	in P [13]	NP-complete [13]	in AC_0 [13]	in AC_0 [13]
$\mathcal{EL}$, $\mathcal{ELH}$	P-complete [6]	NP-complete [87,59]	P-complete [6]	P-complete [87,57,59]
$\mathcal{EL}^+, \mathcal{EL}^{++}$	P-complete [6]	undecidable [87,57,59] (PSPACE-complete for *regular*-$\mathcal{EL}^{++}$ [59])	P-complete [6]	undecidable [87,57,59] (P-complete for *regular*-$\mathcal{EL}^{++}$ [59])
Horn-$\mathcal{SHIQ}$, Horn-$\mathcal{SHOIQ}$	EXPTIME-complete (Horn-$\mathcal{SHIQ}$ [60], Horn-$\mathcal{SHOIQ}$ [77])	EXPTIME-complete (Horn-$\mathcal{SHIQ}$ [29], Horn-$\mathcal{SHOIQ}$ [78])	P-complete (Horn-$\mathcal{SHIQ}$ [53], Horn-$\mathcal{SHOIQ}$ [78])	P-complete (Horn-$\mathcal{SHIQ}$ [60], Horn-$\mathcal{SHOIQ}$ [77])
Horn-$\mathcal{SRIQ}$	in 2EXPTIME [77]	in 2EXPTIME [78]	P-complete [78]	P-complete [78]
Horn-$\mathcal{SROIQ}$	2EXPTIME-complete [77]	2EXPTIME-complete [78]	P-complete [78]	P-complete [78]
$\mathcal{ALC}$, $\mathcal{ALCHQ}$	EXPTIME-complete [92,70]	EXPTIME-complete [66,79]	CONP-complete l.b.[91], u.b.[75]	CONP-complete l.b.[91], u.b.[75]
$\mathcal{ALCI}$, $\mathcal{SH}$, $\mathcal{ALCHIQ}$, $\mathcal{SHIQ}$,$\mathcal{SHOQ}$,$\mathcal{SHIO}$ $\mathcal{ZIQ}$,$\mathcal{ZOQ}$,$\mathcal{ZOI}$	EXPTIME-complete (cf. [95,23] and references therein, see also [7])	2EXPTIME-complete (l.b. $\mathcal{ALCI}$ [66], $\mathcal{SH}$ [31]; u.b. $\mathcal{ZIQ}$, $\mathcal{ZOQ}$, $\mathcal{ZOI}$ [23], $\mathcal{SHIQ}$ [36,22], $\mathcal{SHOQ}$ [37], $\mathcal{ALCHIQ}$ [52], cf. Sect. 8)	CONP-hard [91], CONP-complete for many DLs ($\mathcal{SHIQ}$ [53], $\mathcal{SHOQ}, \mathcal{SHOI}$ [75])	CONP-hard [91], CONP-complete for many DLs ($\mathcal{ALCHIQ}$ [53], $\mathcal{SHIQ}$ [36] $\mathcal{ALCHOQ}, \mathcal{ALCHOI}$ [75])
$\mathcal{SRIQ}$	2EXPTIME-complete l.b.[55],u.b.[23]	in 3EXPTIME [23]	CONP-complete (follows from [85] and [55])	CONP-complete [85]
$\mathcal{ALCHOIQ}$	NEXPTIME-complete [94]	decidable [89], CO-N2EXPTIME-hard [39]	CONP-complete [85]	decidable [89]
$\mathcal{SHOIQ}$	NEXPTIME-complete [94]	open	CONP-complete [85]	open
$\mathcal{SROIQ}$	N2EXPTIME-complete [55]	open	CONP-complete (follows from [85] and [55])	open
$\mathcal{ZOIQ}$	open	undecidable [76]	open	undecidable [76]

counterparts, although exponentially less succinct (the difference in succinctness has only been shown for $\mathcal{SRIQ}$). Saying that query answering in most expressive DLs is in general exponentially harder than standard reasoning gives a rough approximation of the complexity landscape which, as the figure reveals, is not entirely trivial.

9 Summary and Conclusions

In this chapter we have given an introduction to Description Logics and surveyed one of their most vibrant subfields of research: querying data repositories through DL ontologies, also referred to as *Ontology Based Data Access*. Our emphasis was on expressive DLs for which, unfortunately, algorithms remain at the theoretical level and no realization is available. We have also given an overview of the state of the art for lightweight DLs, on which most of the practical work has focused. For quick reference, we close the chapter with a table summarizing the main complexity bounds for both standard reasoning and query answering in both lightweight and expressive DLs. Very roughly, one can say that in the lightweight DLs of the *DL-Lite* and $\mathcal{EL}$ families, standard reasoning is tractable in both data and combined complexity, while conjunctive query answering is tractable in data complexity, but NP-complete in combined complexity. In more expressive Horn fragments the complexity of standard reasoning and of query answering coincides: tractable in data complexity but ExpTime-complete (or 2ExpTime-complete for the $\mathcal{SR}$ family) in combined complexity. For the expressive extensions of $\mathcal{ALC}$ where standard reasoning is ExpTime complete in combined complexity, query answering is usually 2ExpTime complete in combined complexity and coNP complete in data complexity. However, for the NExpTime-hard DLs that support inverses, nominals and counting, the complexity of query answering remains open.

References

1. Abiteboul, S., Buneman, P., Suciu, D.: Data on the Web: from Relations to Semistructured Data and XML. Morgan Kaufmann (2000)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley Publ. Co. (1995)
3. Adali, S., Candan, K.S., Papakonstantinou, Y., Subrahmanian, V.S.: Query caching and optimization in distributed mediator systems. In: Proc. of the ACM SIGMOD Int. Conf. on Management of Data, pp. 137–148 (1996)
4. Artale, A., Calvanese, D., Kontchakov, R., Zakharyaschev, M.: The *DL-Lite* family and relations. J. of Artificial Intelligence Research 36, 1–69 (2009)
5. Baader, F.: Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In: Proc. of the 12th Int. Joint Conf. on Artificial Intelligence, IJCAI 1991 (1991)
6. Baader, F., Brandt, S., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI 2005 (2005)

7. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications, 2nd edn. Cambridge University Press (2007)
8. Baader, F., Hladik, J., Lutz, C., Wolter, F.: From tableaux to automata for description logics. Fundamenta Informaticae 57, 1–33 (2003)
9. Bienvenu, M., Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in the description logic $\mathcal{S}$. In: Proc. of the 23rd International Workshop on Description Logics, DL 2010. CEUR-WS (2010)
10. Bienvenu, M., Ortiz, M., Šimkus, M.: Answering expressive path queries over lightweight DL knowledge bases. In: Description Logics (2012)
11. Bonatti, P., Lutz, C., Murano, A., Vardi, M.Y.: The complexity of enriched μ-calculi. Logical Methods in Computer Science 4(3:11), 1–27 (2008)
12. Buneman, P.: Semistructured data. In: Proc. of the 16th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, PODS 1997, pp. 117–121 (1997)
13. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. of Automated Reasoning 39(3), 385–429 (2007)
14. Calvanese, D., De Giacomo, G., Lenzerini, M.: On the decidability of query containment under constraints. In: Proc. of the 17th ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, PODS 1998, pp. 149–158 (1998)
15. Calvanese, D., De Giacomo, G., Lenzerini, M.: 2ATAs make DLs easy. In: CEUR Electronic Workshop Proceedings of Proc. of the 2002 Description Logic Workshop,DL 2002, pp. 107–118 (2002)
16. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logics constraints. ACM Trans. on Computational Logic 9(3), 22.1–22.31 (2008)
17. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Logical foundations of peer-to-peer data integration. In: Proc. of the 23rd ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, PODS 2004, pp. 241–251 (2004)
18. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Rewriting regular expressions in semi-structured data. In: Proc. of ICDT 1999 Workshop on Query Processing for Semi-Structured Data and Non-Standard Data Formats (1999)
19. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Containment of conjunctive regular path queries with inverse. In: Proc. of the Seventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 176–185 (2000)
20. Calvanese, D., De Giacomo, G., Lenzerini, M., Vardi, M.Y.: Reasoning on regular path queries. SIGMOD Record 32(4), 83–92 (2003)
21. Calvanese, D., De Giacomo, G., Vardi, M.Y.: Decidable containment of recursive queries. Theoretical Computer Science 336(1), 33–56 (2005)
22. Calvanese, D., Eiter, T., Ortiz, M.: Answering regular path queries in expressive description logics: An automata-theoretic approach. In: Proc. of the 22nd AAAI Conference on Artificial Intelligence, AAAI 2007, pp. 391–396 (2007)
23. Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Boutilier, C. (ed.) Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009, pp. 714–720 (2009)

24. Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Proc. of the 9th ACM Symp. on Theory of Computing, STOC 1977, pp. 77–90 (1977)
25. Chortaras, A., Trivela, D., Stamou, G.: Optimized Query Rewriting for OWL 2 QL. In: Bjørner, N., Sofronie-Stokkermans, V. (eds.) CADE 2011. LNCS, vol. 6803, pp. 192–206. Springer, Heidelberg (2011), http://dl.acm.org/citation.cfm?id=2032266.2032282
26. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., Sattler, U.: OWL 2: The next step for OWL. Journal of Web Semantics 6(4), 309–322 (2008)
27. De Giacomo, G.: Decidability of Class-Based Knowledge Representation Formalisms. Ph.D. thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza" (1995)
28. De Giacomo, G., Lenzerini, M.: Boosting the correspondence between description logics and propositional dynamic logics. In: Proc. of the 12th Nat. Conf. on Artificial Intelligence, AAAI 1994, pp. 205–212 (1994)
29. Eiter, T., Gottlob, G., Ortiz, M., Šimkus, M.: Query answering in the description logic horn-$\mathcal{SHIQ}$. In: Hölldobler, S., Lutz, C., Wansing, H. (eds.) JELIA 2008. LNCS (LNAI), vol. 5293, pp. 166–179. Springer, Heidelberg (2008)
30. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query Answering in Description Logics: The Knots Approach. In: Ono, H., Kanazawa, M., de Queiroz, R. (eds.) WoLLIC 2009. LNCS, vol. 5514, pp. 26–36. Springer, Heidelberg (2009)
31. Eiter, T., Lutz, C., Ortiz, M., Šimkus, M.: Query answering in description logics with transitive roles. In: Boutilier, C. (ed.) Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009, pp. 759–764 (2009)
32. Eiter, T., Ortiz, M., Simkus, M.: Conjunctive query answering in the description logic SH using knots. J. Comput. Syst. Sci. 78(1), 47–85 (2012)
33. Eiter, T., Ortiz, M., Šimkus, M., Tran, T.K., Xiao, G.: Query rewriting for Horn-SHIQ plus rules. In: Proc. of the 26th AAAI Conference on Artificial Intelligence, AAAI 2012 (to appear, 2012)
34. Franconi, E.: Description logics for natural language processing. In: Working Notes of the AAAI Fall Symposium on "Knowledge Representation for Natural Language Processing in Implemented Systems", pp. 37–44 (1994)
35. Gehrke, M., Burkert, G., Forster, P., Franconi, E.: Natural language processing and description logics. In: Peltason, C., von Luck, K., Kindermann, C. (eds.) Proc. of the Terminological Logic Users Workshop, pp. 162–164. Department of Computer Science, Technische Universität Berlin (1991)
36. Glimm, B., Horrocks, I., Lutz, C., Sattler, U.: Conjunctive query answering for the description logic $\mathcal{SHIQ}$. Journal of Artificial Intelligence Research 31, 151–198 (2008)
37. Glimm, B., Horrocks, I., Sattler, U.: Unions of conjunctive queries in SHOQ. In: Proc. of the 11th International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008), pp. 252–262. AAAI Press/The MIT Press (2008)
38. Glimm, B., Kazakov, Y.: Role Conjunctions in Expressive Description Logics. In: Cervesato, I., Veith, H., Voronkov, A. (eds.) LPAR 2008. LNCS (LNAI), vol. 5330, pp. 391–405. Springer, Heidelberg (2008)
39. Glimm, B., Kazakov, Y., Lutz, C.: Status QIO: An update. In: Proc. of the 2009 Description Logic Workshop, DL 2009. CEUR Workshop Proceedings, vol. 745 (2011)

40. Gottlob, G., Orsi, G., Pieris, A.: Ontological queries: Rewriting and optimization. In: IEEE 27th International Conference on Data Engineering, ICDE, pp. 2–13 (April 2011)
41. Gottlob, G., Orsi, G., Pieris, A., Šimkus, M.: Datalog and its Extensions for the Semantic Web Databases. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012, vol. 7487, pp. 54–77. Springer, Heidelberg (2012)
42. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 745, CEUR-WS.org (2011)
43. Grädel, E.: Why are modal logics so robustly decidable? Bulletin of the European Association for Theoretical Computer Science 68, 90–103 (1999)
44. Gupta, A., Ullman, J.D.: Generalizing conjunctive query containment for view maintenance and integrity constraint verification (abstract). In: Workshop on Deductive Databases (In conjunction with JICSLP), Washington D.C. (USA), p. 195 (1992)
45. Hladik, J.: A tableau system for the description logic SHIO. In: Sattler, U. (ed.) IJCAR Doctoral Programme. CEUR Workshop Proceedings, vol. 106, CEUR-WS.org (2004)
46. Horrocks, I., Kutz, O., Sattler, U.: The irresistible $\mathcal{SRIQ}$. In: Proc. of the 1st Int. Workshop on OWL: Experiences and Directions, OWLED 2005 (2005)
47. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible $\mathcal{SROIQ}$. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press (2006)
48. Horrocks, I., Rector, A., Goble, C.: A description logic based schema for the classification of medical data. In: CEUR Electronic Workshop Proceedings, Proc. of the 3rd Int. Workshop on Knowledge Representation Meets Databases, KRDB 1996, pp. 24–28 (1996), http://ceur-ws.org/Vol-4/
49. Horrocks, I., Sattler, U., Tobies, S.: Practical reasoning for very expressive description logics. J. of the Interest Group in Pure and Applied Logic 8(3), 239–264 (2000)
50. Horrocks, I., Sattler, U., Tobies, S.: Reasoning with Individuals for the Description Logic $\mathcal{SHIQ}$. In: McAllester, D. (ed.) CADE 2000. LNCS, vol. 1831, pp. 482–496. Springer, Heidelberg (2000)
51. Horrocks, I., Tessaris, S.: A conjunctive query language for description logic ABoxes. In: Proc. of the 17th Nat. Conf. on Artificial Intelligence, AAAI 2000, pp. 399–404 (2000)
52. Hustadt, U., Motik, B., Sattler, U.: A Decomposition Rule for Decision Procedures by Resolution-Based Calculi. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS (LNAI), vol. 3452, pp. 21–35. Springer, Heidelberg (2005)
53. Hustadt, U., Motik, B., Sattler, U.: Data complexity of reasoning in very expressive description logics. In: Proc. of the 19th Int. Joint Conf. on Artificial Intelligence, IJCAI 2005, pp. 466–471 (2005)
54. Immerman, N.: Expressibility and parallel complexity. SIAM J. Comput. 18(3), 625–638 (1989)
55. Kazakov, Y.: $\mathcal{RIQ}$ and $\mathcal{SROIQ}$ are harder than $\mathcal{SHOIQ}$. In: Proc. of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning, KR 2008, pp. 274–284 (2008)

56. Kikot, S., Kontchakov, R., Zakharyaschev, M.: On (in)tractability of OBDA with OWL 2 QL. In: Rosati, R., Rudolph, S., Zakharyaschev, M. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 745, CEUR-WS.org (2011), http://dblp.uni-trier.de/db/conf/dlog/dlog2011.html#KikotKZ11
57. Krisnadhi, A., Lutz, C.: Data Complexity in the $\mathcal{EL}$ Family of Description Logics. In: Dershowitz, N., Voronkov, A. (eds.) LPAR 2007. LNCS (LNAI), vol. 4790, pp. 333–347. Springer, Heidelberg (2007)
58. Krötzsch, M.: OWL 2 profiles: An Introduction to Lightweight Ontology Languages. In: Eiter, T., Krennwallner, T. (eds.) Reasoning Web 2012. LNCS, vol. 7487, Springer, Heidelberg (2012)
59. Krötzsch, M., Rudolph, S.: Conjunctive queries for $\mathcal{EL}$ with composition of roles. In: Proc. of the 2007 Description Logic Workshop, DL 2007. CEUR Electronic Workshop Proceedings, vol. 250 (2007), http://ceur-ws.org/Vol-250/
60. Krötzsch, M., Rudolph, S., Hitzler, P.: Complexity boundaries for Horn description logics. In: Proceedings of the 22nd AAAI Conference on Artificial Intelligence, AAAI 2007, pp. 452–457. AAAI Press (2007)
61. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems, PODS 2002, pp. 233–246 (2002)
62. Levy, A.Y., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artificial Intelligence 104(1-2), 165–209 (1998)
63. Lutz, C., Toman, D., Wolter, F.: Conjunctive query answering in the description logic $\mathcal{EL}$ using a relational database system. In: Proc. of the 21st Int. Joint Conf. on Artificial Intelligence, IJCAI 2009, pp. 2070–2075. AAAI Press (2009)
64. Lutz, C.: Complexity of Terminological Reasoning Revisited. In: Ganzinger, H., McAllester, D., Voronkov, A. (eds.) LPAR 1999. LNCS, vol. 1705, pp. 181–200. Springer, Heidelberg (1999)
65. Lutz, C.: Inverse roles make conjunctive queries hard. In: Proc. of the 2007 Description Logic Workshop, DL 2007. CEUR Electronic Workshop Proceedings, vol. 250, pp. 100–111 (2007), http://ceur-ws.org/Vol-250/
66. Lutz, C.: The Complexity of Conjunctive Query Answering in Expressive Description Logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
67. Lutz, C.: Two upper bounds for conjunctive query answering in SHIQ. In: Baader, F., Lutz, C., Motik, B. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 353, CEUR-WS.org (2008)
68. McGuinness, D.L.: Ontology-enhanced search for primary care medical literature. In: Proc. of the Int. Medical Informatics Association Working Group 6 – Conference on Natural Language Processing and Medical Concept Representation, IMIA 1999 (1999)
69. Motik, B.: Reasoning in Description Logics using Resolution and Deductive Databases. Ph.D. thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany (January 2006)
70. Nebel, B.: Terminological reasoning is inherently intractable. Artificial Intelligence 43, 235–249 (1990)
71. Németi, I.: Free algebras and decidability in algebraic logic. DSc. thesis, Mathematical Institute of The Hungarian Academy of Sciences, Budapest (1986)
72. Noy, N.F.: Semantic integration: A survey of ontology-based approaches. SIGMOD Record 33(4), 65–70 (2004)
73. Ortiz, M.: Query Answering in Expressive Description Logics: Techniques and Complexity Results. Ph.D. thesis, Vienna University of Technology (2010)

74. Ortiz, M., Calvanese, D., Eiter, T.: Characterizing data complexity for conjunctive query answering in expressive description logics. In: Proc. of the 21st Nat. Conf. on Artificial Intelligence, AAAI 2006. AAAI Press (July 2006)
75. Ortiz, M., Calvanese, D., Eiter, T.: Data complexity of query answering in expressive description logics via tableaux. J. of Automated Reasoning 41(1), 61–98 (2008)
76. Ortiz, M., Rudolph, S., Šimkus, M.: Query answering is undecidable in DLs with regular expressions, inverses, nominals, and counting. Tech. Rep. INFSYS RR-1843-10-03, Institut für Informationssysteme, Technische Universität Wien, A-1040 Vienna, Austria (April 2010)
77. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Lin, F., Sattler, U., Truszczynski, M. (eds.) KR 2010, AAAI Press (2010)
78. Ortiz, M., Rudolph, S., Simkus, M.: Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In: Walsh, T. (ed.) IJCAI, pp. 1039–1044. IJCAI/AAAI (2011)
79. Ortiz, M., Šimkus, M., Eiter, T.: Worst-case optimal conjunctive query answering for an expressive description logic without inverses. In: Fox, D., Gomes, C.P. (eds.) AAAI 2008, pp. 504–510. AAAI Press (2008)
80. Patel-Schneider, P., Hayes, P., Horrocks, I.: OWL Web Ontology Language semantics and abstract syntax – W3C recommendation. Tech. rep., World Wide Web Consortium (February 2004), http://www.w3.org/TR/owl-semantics/
81. Pérez-Urbina, H., Motik, B., Horrocks, I.: A comparison of query rewriting techniques for DL-Lite. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) Description Logics. CEUR Workshop Proceedings, vol. 477, CEUR-WS.org (2009)
82. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. J. Applied Logic 8(2), 186–209 (2010)
83. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking Data to Ontologies. In: Spaccapietra, S. (ed.) Journal on Data Semantics X. LNCS, vol. 4900, pp. 133–173. Springer, Heidelberg (2008)
84. Pratt, V.R.: Models of program logic. In: Proc. of the 20th Annual Symp. on the Foundations of Computer Science (FOCS 1979), pp. 115–122 (1979)
85. Pratt-Hartmann, I.: Data-complexity of the two-variable fragment with counting quantifiers. Information and Computation 207(8), 867–888 (2009)
86. Rector, A., Bechhofer, S., Goble, C.A., Horrocks, I., Nowlan, W.A., Solomon, W.D.: The GRAIL concept modelling language for medical terminology. Artificial Intelligence in Medicine 9, 139–171 (1997)
87. Rosati, R.: On conjunctive query answering in $\mathcal{EL}$. In: Proc. of the 2007 Description Logic Workshop, DL 2007. CEUR Electronic Workshop Proceedings, vol. 250 (2007), http://ceur-ws.org/Vol-250/
88. Rosati, R., Almatelli, A.: Improving query answering over DL-Lite ontologies. In: Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, KR 2010 (2010)
89. Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! J. of Artificial Intelligence Research 39, 429–481 (2010)
90. Sattler, U., Vardi, M.Y.: The hybrid μ-calculus. In: Proc. of the Int. Joint Conf. on Automated Reasoning, IJCAR 2001, pp. 76–91 (2001)
91. Schaerf, A.: On the complexity of the instance checking problem in concept languages with existential quantification. J. of Intelligent Information Systems 2, 265–278 (1993)

92. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. of the 12th Int. Joint Conf. on Artificial Intelligence, IJCAI 1991, pp. 466–471 (1991)
93. Tessaris, S.: Questions and Answers: Reasoning and Querying in Description Logic. Ph.D. thesis, University of Manchester, Department of Computer Science (April 2001)
94. Tobies, S.: The complexity of reasoning with cardinality restrictions and nominals in expressive description logics. J. of Artificial Intelligence Research 12, 199–217 (2000)
95. Tobies, S.: Complexity Results and Practical Algorithms for Logics in Knowledge Representation. Ph.D. thesis, LuFG Theoretical Computer Science, RWTH-Aachen, Germany (2001)
96. Ullman, J.D.: Information integration using logical views. Theoretical Computer Science 239(2), 189–210 (2000)
97. Vardi, M.Y.: The complexity of relational query languages. In: Proc. of the 14th ACM SIGACT Symp. on Theory of Computing, STOC 1982, pp. 137–146 (1982)
98. Vardi, M.Y.: Reasoning about the Past with Two-Way Automata. In: Larsen, K.G., Skyum, S., Winskel, G. (eds.) ICALP 1998. LNCS, vol. 1443, pp. 628–641. Springer, Heidelberg (1998)
99. Vardi, M.Y., Wolper, P.: Automata-theoretic techniques for modal logics of programs. Journal of Computer and System Sciences 32, 183–221 (1986)
100. Vollmer, H.: Introduction to circuit complexity - a uniform approach. In: Texts in Theoretical Computer Science. Springer (1999)