Description Logics

Markus Krötzsch, František Simančík, and Ian Horrocks

Department of Computer Science, University of Oxford, UK

Abstract. This article provides a self-contained first introduction to description logics (DLs). The main concepts and features are explained with examples before the syntax and semantics of the DL $\mathcal{SROIQ}$ are defined in detail. Additional sections review lightweight DL languages, discuss the relationship to the OWL Web Ontology Language and give pointers to further reading.

1 Introduction

Description logics (DLs) are a family of knowledge representation languages that are widely used in ontological modelling. An important practical reason for this is that they provide one of the main underpinnings for the OWL Web Ontology Language as standardised by the World Wide Web Consortium (W3C). However, DLs have been used in knowledge representation long before the advent of ontological modelling in the context of the Semantic Web, tracing back to first DL modelling languages in the mid 1980s.

As their name suggests, DLs are logics (in fact most DLs are decidable fragments of first-order logic), and as such they are equipped with a *formal semantics*: a precise specification of the meaning of DL ontologies. This formal semantics allows humans and computer systems to exchange DL ontologies without ambiguity as to their meaning, and also makes it possible to use logical deduction to *infer* additional information from the facts stated explicitly in an ontology—an important feature that distinguishes DLs from other modelling languages such as UML.

The capability of inferring additional knowledge increases the modelling power of DLs but it also requires some understanding on the side of the modeller and, above all, good tool support for computing the conclusions. The computation of inferences is called *reasoning* and an important goal of DL language design has been to ensure the availability of practical reasoning algorithms. This is one of the reasons why there is not just a single description logic: the best balance between expressivity of the language and complexity of reasoning depends on the intended application.

In this article we provide a self-contained first introduction to description logics, including basic features and ideas, and give pointers to several advanced topics.

2 Basic Building Blocks of DL Ontologies

DLs provide the means to model relationships between individuals in a domain of interest. DL ontologies are based on three kinds of building blocks: *concepts* represent sets of individuals, *roles* represent binary relations between the individuals, and *individual*

names represent single individuals in the domain. Readers familiar with first-order logic will recognise these as unary predicates, binary predicates and constants.

For example, an ontology modelling family relationships might use concepts such as Mother to represent the set of all mothers, roles such as parentOf to represent the (binary) relationship between parents and their children, and individual names such as julia to represent the individual Julia.

Unlike a database, a DL ontology does not fully describe a particular situation or "state of the world"; rather it consists of a set of statements, called axioms, which typically capture only partial knowledge about the described situation. Thus there may be many different states of the world that are consistent with the ontology. Although there is no principal logical difference between different types of axioms, they are often separated into three groups: assertional (ABox) axioms, terminological (TBox) axioms and relational (RBox) axioms.

We continue with an intuitive introduction to the most important DL modelling features, starting with the ABox (Section 3), and followed by TBox (Section 4) and RBox (Section 5). This leads us to the rather expressive DL called $\mathcal{SROIQ}$, the syntax of which we summarise in Section 6. In Section 7 we explain the underlying ideas of DL semantics and use it to define the meaning of $\mathcal{SROIQ}$ ontologies. In Section 8 we review several practically important lightweight DLs that can be obtained by omitting some features of $\mathcal{SROIQ}$. In Section 9 we discuss the relationship of DLs to the OWL Web Ontology Language. We conclude with pointers to further reading in Section 10.

3 Asserting Facts with ABox Axioms

ABox axioms capture knowledge about named individuals, especially the concepts they belong to and their mutual relationships. The most common ABox axioms are *concept assertions* such as

$$\mathsf{Mother}(\mathsf{julia}), \tag{1}$$

which asserts that Julia is a mother or, more precisely, that the individual named julia is an *instance* of the concept Mother.

Role assertions describe relations between named individuals. The assertion

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}), \tag{2}$$

for example, states that Julia is a parent of John or, more precisely, that the individual named julia is in the relation that is represented by parentOf to the individual named john. It is often cumbersome to emphasise that the relationships expressed by axioms are really relationships between the individuals, sets and relations that are represented by the respective individual names, concepts and roles. Assuming that the distinction between syntactic identifiers and semantic entities is understood, we often prefer more sloppy and readable formulations. Section 7 explains the underlying semantics with greater precision.

Intuitively, Julia and John are different individuals, but this does not logically follow from the previous axioms. Most DLs do not make the *unique name assumption*, so

different names might refer to the same individual unless explicitly stated otherwise. The *individual inequality* assertion

$$\mathsf{julia} \not\approx \mathsf{john} \tag{3}$$

is used to assert that Julia and John are different individuals. Conversely, an *individual equality* assertion, such as

$$\mathsf{john} \approx \mathsf{johnny}, \tag{4}$$

states that two different names refer to the same individual. This can arise, for example, when integrating information from several sources.

4 Expressing Terminological Knowledge with TBox Axioms

TBox axioms describe relationships between concepts. For example, the fact that all mothers are parents is expressed by the *concept inclusion*

$$\mathsf{Mother} \sqsubseteq \mathsf{Parent}, \tag{5}$$

in which case we say that the concept Mother is *subsumed* by the concept Parent. Such knowledge allows us to infer further facts about individuals. For example, (1) and (5) together imply that Julia is a parent. *Concept equivalence* asserts that two concepts have the same instances, as in

$$\mathsf{Person} \equiv \mathsf{Human}. \tag{6}$$

While synonyms are an obvious example of equivalent concepts, it is more common to use concept equivalence to give names to complex expressions, which we will introduce next. Indeed, the basic types of axioms introduced so far are rather limited. To describe more complicated situations, DLs allow new concepts to be built using various concept constructors. The resulting complex concept expressions can be used in all places where concept names are allowed. This enables us to describe relationships such as concept disjointness, which asserts that two concepts do not share any instances.

4.1 Boolean Concept Constructors

Boolean concept constructors represent Boolean operations closely related to intersection, union and complement of sets, or to conjunction, disjunction and negation of logical formulae. For example, concept inclusions allow us to state that all mothers are female and that all mothers are parents, but what we really mean is that mothers are *exactly* the female parents. We can achieve this by constructing a new complex concept as the *intersection* (also called *conjunction*)

$$\mathsf{Female} \sqcap \mathsf{Parent}, \tag{7}$$

which represents the set of individuals that are both female and parents. Complex concepts can be used in axioms just like atomic concepts, e.g., in the equivalence $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$.

Union (also called *disjunction*) is the dual of intersection. For example,

$$\mathsf{Father} \sqcup \mathsf{Mother} \tag{8}$$

is the concept of individuals that are fathers or mothers. For example, the axiom $\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$ states that a parent is a father or a mother (and vice versa).

Sometimes we are interested in individuals that do *not* belong to a certain concept, e.g., in women who are not married, which could be described by the complex concept

$$\mathsf{Female} \sqcap \neg\mathsf{Married}, \tag{9}$$

where the *complement* (also called *negation*) $\neg\mathsf{Married}$ represents the set of all unmarried individuals.

It is sometimes useful to make a statement about every individual, e.g., to say that everybody is either male or female. This can be accomplished by the axiom

$$\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \tag{10}$$

where the *top concept* $\top$ is a special concept with every individual as an instance. Axiom (10) is not very accurate, as it requires that every individual has a gender, which may not be reasonable for instances of a concept such as Computer. We will see more useful applications for $\top$ later on.

To express that, in our example, nobody can be both male and female at once, we can declare the two concepts to be *disjoint*. While ontology languages like OWL provide a basic constructor for disjointness, DLs naturally capture it with the axiom

$$\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot, \tag{11}$$

where the *bottom concept* $\bot$ is the dual of $\top$, that is the special concept without any instances. The above axiom thus says that the intersection of the two concepts is empty.

4.2 Role Restrictions

The most interesting feature of DLs is their ability to form statements that link concepts and roles together. For example, there is an obvious relationship between the concept Parent and the role parentOf: a parent is someone who is a parent of at least one individual. In DLs, this relationship can be captured by the concept equivalence

$$\mathsf{Parent} \equiv \exists\mathsf{parentOf}.\top, \tag{12}$$

where the *existential restriction* $\exists\mathsf{parentOf}.\top$ is a complex concept that describes the set of individuals that are parents of at least one individual (instance of $\top$). Similarly, the concept $\exists\mathsf{parentOf}.\mathsf{Female}$ describes individuals that are parents of at least one female individual, i.e., those that have a daughter.

To represent the set of individuals all of whose children are female, we use the *universal restriction*

$$\forall\mathsf{parentOf}.\mathsf{Female}. \tag{13}$$

It is a common error to forget that (13) also includes those individuals without any children. More accurately (and less naturally), the axiom can be said to describe the set of all individuals that have "no children other than female ones," i.e., that have "no children that are not female." Indeed, the concept (13) could be equivalently expressed as $\neg\exists \mathsf{parentOf}.\neg\mathsf{Female}$. If this meaning is not intended, one can describe the individuals who have at least one child and with all their children being female by the concept $(\exists \mathsf{parentOf}.\top) \sqcap (\forall \mathsf{parentOf}.\mathsf{Female})$.

Existential and universal restrictions together with the top concept can express *domain* and *range restrictions* on roles. For example, we can restrict the domain and range of the role sonOf to male individuals and to parents, respectively:

$$\exists \mathsf{sonOf}.\top \sqsubseteq \mathsf{Male}, \tag{14}$$

$$\top \sqsubseteq \forall \mathsf{sonOf}.\mathsf{Parent}. \tag{15}$$

Together with the assertion $\mathsf{sonOf}(\mathsf{john}, \mathsf{julia})$, these axioms would allow us to deduce that John is male and Julia is a parent. We remark that this differs from the meaning of *constraints* in databases, which are used to check the validity of given data ("all sons must be male") rather than being used to deduce new data. Mistaking DL axioms for constraints is a very common source of modelling errors.

Number restrictions allow us to restrict the number of individuals that are reachable via some role. For example, we can describe the set of individuals that are children of at least two parents ($\geqslant 2\, \mathsf{childOf}.\mathsf{Parent}$) or at most two parents ($\leqslant 2\, \mathsf{childOf}.\mathsf{Parent}$). The axiom $\mathsf{Person} \sqsubseteq \geqslant 2\, \mathsf{childOf}.\mathsf{Parent} \sqcap \leqslant 2\, \mathsf{childOf}.\mathsf{Parent}$ then states that every person is a child of exactly two parents.

Finally, *local reflexivity* can be used to describe the set of individuals that are related to themselves via a given role. For example, the set of individuals that talk to themselves is described by the concept

$$\exists \mathsf{talksTo}.\mathit{Self}. \tag{16}$$

4.3 Nominals

It may also be useful to define a concept by simply enumerating its instances. Unlike in OWL, there is no DL construct for enumerations, but we can express them using *nominals*. A nominal is a concept that has exactly one instance. For example, $\{\mathsf{john}\}$ is the concept whose only instance is (the individual represented by) john. Enumerations can thus be expressed as unions of nominals, as in the next example:

$$\mathsf{Beatle} \equiv \{\mathsf{john}\} \sqcup \{\mathsf{paul}\} \sqcup \{\mathsf{george}\} \sqcup \{\mathsf{ringo}\}. \tag{17}$$

Using nominals, a concept assertion $\mathsf{Mother}(\mathsf{julia})$ can be turned into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \mathsf{Mother}$ and a role assertion $\mathsf{parentOf}(\mathsf{julia}, \mathsf{john})$ into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \exists \mathsf{parentOf}.\{\mathsf{john}\}$. This shows that ABox axioms could also be expressed as part of the TBox. The distinction between ABox and TBox is nonetheless useful for modelling purposes, separating general domain knowledge (TBox) from specific application data (ABox).

5 Characterising Roles with RBox Axioms

RBox axioms refer to properties of roles. As for concepts, DLs support *role inclusion* and *role equivalence* axioms. For example, the inclusion

$$\mathsf{parentOf} \sqsubseteq \mathsf{ancestorOf} \tag{18}$$

states that parentOf is a *subrole* of ancestorOf: every pair of individuals related by parentOf is also related by ancestorOf. Thus (2) and (18) together imply that Julia is an ancestor of John.

In role inclusion axioms, *role composition* can be used to describe roles such as uncleOf. Intuitively, if Charles is a brother of Julia and Julia is a parent of John, then Charles is an uncle of John. This kind of relationship between the roles brotherOf, parentOf and uncleOf is captured by the *complex role inclusion* axiom

$$\mathsf{brotherOf} \circ \mathsf{parentOf} \sqsubseteq \mathsf{uncleOf}. \tag{19}$$

Role composition can only appear on the left-hand side of complex role inclusions. Many DLs impose some additional restrictions that determine if a collection of such axioms can be used together in one ontology.

Nobody can be both a parent and a child of the same individual, so the two roles parentOf and childOf are *disjoint*, which we can write as follows:

$$\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{childOf}). \tag{20}$$

In contrast to the variety of concept constructors, DLs provide only few constructors for forming complex roles. In practice, *inverse roles* are the most important such constructor. For example, parentOf is the inverse of childOf: if Julia is a parent of John, then John is a child of Julia and vice versa. This can be expressed by the axiom

$$\mathsf{parentOf} \equiv \mathsf{childOf}^{-}, \tag{21}$$

where the complex role $\mathsf{childOf}^{-}$ represents the inverse of childOf. Inverse roles can be used in all places where non-inverse roles can be used.

Moreover, DLs also provide a special *universal role U* that relates all pairs of individuals, and (rarely) an *empty role* that relates no individuals. Both are rarely used in modelling and are mainly provided for symmetry with the top and bottom concepts.

OWL provides a variety of other RBox axioms, namely role transitivity, symmetry, asymmetry, reflexivity and irreflexivity. These are sometimes considered as basic axiom types in DLs as well, using some suggestive notation such as *Trans*(ancestorOf) to express that the role ancestorOf is transitive. However, such axioms are just syntactic sugar; all role characteristics can be expressed using features that we have already introduced.

Transitivity is a special form of complex role inclusion. For example, transitivity of ancestorOf can be captured by the axiom $\mathsf{ancestorOf} \circ \mathsf{ancestorOf} \sqsubseteq \mathsf{ancestorOf}$. A role is *symmetric* if it is equivalent to its inverse, e.g., $\mathsf{marriedTo} \equiv \mathsf{marriedTo}^{-}$, and it is *asymmetric* if it is disjoint from its inverse, as in $\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{parentOf}^{-})$. If desired, *global reflexivity* can be expressed by imposing local reflexivity on the top concept as in $\top \sqsubseteq \exists \mathsf{knows}.\mathit{Self}$. A role is *irreflexive* if it is never locally reflexive, as in the case of $\top \sqsubseteq \neg \exists \mathsf{marriedTo}.\mathit{Self}$.

6 The Description Logic $\mathcal{SROIQ}$

Next, we summarise the features introduced in the previous sections to obtain a comprehensive definition of DL syntax. Doing so yields the DL called $\mathcal{SROIQ}$, one of the most expressive DLs considered today, which is closely related to the ontology language OWL 2 DL.

Formally, every DL ontology is based on three finite sets of signature symbols: a set N_I of *individual names*, a set N_C of *concept names* and a set N_R of *role names*. Usually these sets are assumed to be fixed for some application and are therefore not mentioned explicitly. A $\mathcal{SROIQ}$ *role expression* over this signature is a role name, the inverse of a role name, or the special symbol U (universal role). $\mathcal{SROIQ}$ *concept expressions* are defined recursively. Every concept name, $\top$, and $\bot$ is a concept expression, and if C and D are concept expressions, then so are $(C \sqcap D)$, $(C \sqcup D)$, $\neg C$, $\exists R.C$, $\forall R.C$, $\geqslant n\, R.C$, $\leqslant n\, R.C$, $\exists R.\mathit{Self}$, and $\{a\}$, where R is a role expression, $n \geq 0$ is an integer, and $a \in \mathsf{N}_I$. We omit parentheses if there is no semantic confusion. For example, parentheses do not matter for $A \sqcup B \sqcup C$, whereas the expression $A \sqcap B \sqcup C$ is ambiguous.

Axioms are built from concept expressions, role expressions and individual names. ABox axioms are of the form $C(a)$, $R(a, b)$, $a \approx b$, or $a \not\approx b$; TBox axioms are of the form $C \sqsubseteq D$ or $C \equiv D$; RBox axioms are of the form $R \sqsubseteq T$, $R \equiv T$, $R \circ S \sqsubseteq T$, or $\mathit{Disjoint}(R, S)$. Note that both C and D can be complex concept expressions.

Roughly speaking, a $\mathcal{SROIQ}$ ontology (or *knowledge base*) is simply a set of such axioms. To ensure the existence of reasoning algorithms that are correct and terminating, however, additional syntactic restrictions must be imposed. These refer not to single axioms but to the structure of the ontology as a whole, hence they are called *structural restrictions*. This also means that the union of two ontologies that satisfy these restrictions may no longer do so, which must be taken into account when merging ontologies. The two such conditions relevant for $\mathcal{SROIQ}$ are based on the notions of *simplicity* and *regularity*. Roughly speaking, simplicity requires that roles in number restrictions do not depend on complex role inclusion axioms while regularity forbids some forms of cyclic dependencies between complex role inclusion axioms. Both are automatically satisfied for ontologies that do not contain complex role inclusion axioms. Formal definitions of both conditions can be found in the literature [9].

7 Description Logic Semantics

The formal meaning of DL axioms is given by their model-theoretic semantics. In particular, the semantics specifies what the logical consequences of an ontology are. The formal semantics is therefore the main guideline for every tool that computes logical consequences of DL ontologies, and a basic understanding of its working is vital to make reasonable modelling choices and to comprehend the results given by software applications. Luckily, the semantics of DLs is not difficult to understand provided that some common misconceptions are avoided.

Intuitively speaking, an ontology describes a particular situation in a domain of discourse. For example, the axioms in Sections 3–5 describe a particular situation in the "families and relationships" domain. However, ontologies usually cannot fully specify

the situation that they describe. On the one hand, there is no formal relationship between the symbols we use and the objects that they represent: the individual name julia, for example, is just a syntactic identifier with no intrinsic meaning. Indeed, the intended meaning of the identifiers in ontologies has no influence on their formal semantics: what we know about them stems only from ontological axioms. On the other hand, the axioms in an ontology typically do not provide complete information.

DLs have been designed to handle such incomplete information. Rather than making default assumptions in order to fully specify one particular interpretation for each ontology, the DL semantics generally considers all the possible situations (i.e., states of the world) where the axioms of an ontology would hold (we also say: where the axioms are *satisfied*). This is sometimes called the *Open World Assumption* since it keeps unspecified information open. A logical consequence of an ontology is an axiom that holds in all interpretations that satisfy the ontology, i.e., something that is true in all conceivable states of the world that agree with the ontology. The more axioms an ontology contains, the fewer interpretations exist that satisfy all the axioms, and the fewer interpretations satisfy an ontology, the more logical consequences follow from it. In other words, DL semantics is *monotonic*: additional axioms always lead to additional consequences, or, more informally, the more knowledge we feed into a DL system the more results it returns.

An extreme case is when an ontology is not satisfied in any interpretation. The ontology is then called *unsatisfiable* or *inconsistent*, and *every* axiom holds in all of the (zero) interpretations satisfying the ontology. Such an ontology is clearly not useful, and avoiding inconsistency (and checking for it in the first place) is therefore important during modelling.

To complete our presentation of the DL semantics, we need to clarify what we mean by an "interpretation" and which conditions must hold for it to satisfy some axiom. We closely follow the intuitive ideas explained above: an interpretation $\mathcal{I}$ is given by a set $\Delta^{\mathcal{I}}$ (its *domain*) and an interpretation function $\cdot^{\mathcal{I}}$ that maps atomic concepts A to sets $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, atomic roles R to binary relations $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and individual names a to elements $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation of complex concepts and roles follows from the interpretation of the basic entities. Inverse roles are interpreted as inverse relations, i.e., $(R^-)^{\mathcal{I}} = \{\langle y, x\rangle \mid \langle x, y\rangle \in R^{\mathcal{I}}\}$. Boolean operators correspond to set functions: $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$. The semantics of $\exists R.C/\forall R.C/\geqslant n\, R.C/\leqslant n\, R.C$ is the set of all individuals x such that some/all/at least n/at most n $R^{\mathcal{I}}$-successors of x are in $C^{\mathcal{I}}$, where an "$R^{\mathcal{I}}$-successor of x" is any individual y with $\langle x, y\rangle \in R^{\mathcal{I}}$. Finally, $\top$ and $\bot$ have fixed interpretations $\Delta^{\mathcal{I}}$ and $\emptyset$, respectively.

It is now easy to interpret axioms as conditions on an interpretation. For example, a concept inclusion $C \sqsubseteq D$ *holds* in $\mathcal{I}$ if the corresponding set inclusion $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ holds. In this case, we also say that $\mathcal{I}$ *satisfies* $C \sqsubseteq D$ and we write $\mathcal{I} \models C \sqsubseteq D$. It is straightforward to define what it means for other types of axioms to hold; details can be found in the literature [9]. If $\mathcal{I}$ satisfies all axioms in an ontology $\mathcal{O}$, then $\mathcal{I}$ is a *model* of $\mathcal{O}$. Thus a model is an abstraction of a state of the world where all axioms in the ontology hold. An ontology is *consistent* if it has at least one model. An axiom is a *consequence* of an ontology $\mathcal{O}$ if it holds in every model of $\mathcal{O}$. In particular, an inconsistent ontology entails every axiom.

This semantics affects the meaning of individual names in DL ontologies. We already remarked that DLs do not usually make the Unique Name Assumption, and indeed our formal definition allows two individual names to be interpreted as the same individual (element of the domain). Moreover, the domain of an interpretation can contain individuals that are not represented by any individual name. A common confusion in modelling arises from the assumption that interpretations only contain individuals that are represented by individual names (such individuals are also called *named individuals*). For example, one could wrongly assume the ontology consisting of the axioms

$$\mathsf{parentOf}(\mathsf{julia}, \mathsf{john}) \qquad \mathsf{manyChildren}(\mathsf{julia}) \qquad \mathsf{manyChildren} \sqsubseteq \geqslant 3\, \mathsf{parentOf}.\top$$

to be inconsistent since it requires Julia to have at least three children when only one (John) is given. However, there are many models where Julia does have three children, although only one of the children is explicitly named. Many modelling errors can be traced back to similar misconceptions that are easy to prevent if the open world assumption is kept in mind.

Also note that the specification of the semantics does not provide any hint as to how to compute the relevant entailments in practice. There are infinitely many possible interpretations, each of which may have an infinite domain. Therefore it is impossible to test all interpretations to see if they model a given ontology, and impossible to test all models of an ontology to see if they entail a given axiom. Rather, one has to devise deduction procedures and prove their correctness with respect to the above specification. The interplay of certain expressive features can make reasoning algorithms more complicated and in some cases it can even be shown that no correct and terminating algorithm exists at all (i.e., that reasoning is undecidable). However, entailment of axioms is decidable for $\mathcal{SROIQ}$ and a number of free and commercial tools are available. Such tools are typically optimised for more specific reasoning tasks, such as consistency checking, the entailment of concept subsumptions (subsumption checking) or of concept assertions (instance checking). Many standard reasoning tasks are closely related and can be handled by similar algorithms.

8 Lightweight Description Logics

Many DLs have been proposed. Most can be characterised by the types of constructors and axioms that they allow, which are often a subset of those in $\mathcal{SROIQ}$. Since $\mathcal{SROIQ}$ reasoning requires algorithms of very high worst-case complexities, several DLs have been developed in order to allow for more efficient reasoning algorithms. The three main "families" of such lightweight DLs are $\mathcal{EL}$ [1], *DLP* and *DL-Lite* [3], which correspond to language fragments OWL EL, OWL RL and OWL QL of the Web Ontology Language, respectively. Here, we give only a short introduction to this important field and refer to the literature for a more detailed first introduction [8].

The $\mathcal{EL}$ family of DLs is based on existential quantifiers and concept intersection. In addition, one can allow for $\top$ and $\bot$, *Self*, nominals, the universal role, and most types of axioms. Unions, complements, universal and counting quantifiers, and inverse roles are forbidden. All standard reasoning tasks for $\mathcal{EL}$ can be solved in polynomial time. $\mathcal{EL}$ is used to model large but lightweight ontologies, especially in the life sciences. Several

DL reasoners are optimised for handling $\mathcal{EL}$-type ontologies, the most prominent of which currently is the ELK reasoner for OWL EL.

DLP is short for *Description Logic Programs* and supports DL axioms that can be read as rules in first-order Horn logic without function symbols. Thus DLP can be considered as a kind of rule language—hence the name OWL RL. This requires different syntactic restrictions for subconcepts (concepts on the left-hand side of concept inclusion axioms) and superconcepts (those on the right-hand side); we omit the details. To reason with DLP, one can restrict attention to models that contain only domain elements corresponding to some individual name in the ontology. This is why DLP is often used to augment databases (interpreted as sets of ABox axioms), e.g., in an implementation of OWL RL in Oracle 11g.

DL-Lite is a family of DLs that is also used in combination with existing databases, in particular to augment the expressivity of query languages. This approach, known as Ontology-Based Data Access (OBDA), considers ontologies as a language for constructing *views* on top of existing data. OBDA with DL-Lite can be realised with standard query languages such as SQL that are not aware of the DL semantics, where ontologies are used in a query preprocessing step only. Like DLP, DL-Lite requires different syntactic restrictions for subconcepts and superconcepts, which we omit here.

9 Relationship to OWL

The *OWL Web Ontology Language* is a knowledge representation language standardised by the World Wide Web Consortium (W3C). OWL is one of the most important applications of DLs today. We briefly outline the relationship of the two languages. A comprehensive treatment is beyond the scope of this article; see Section 10 for pointers to further reading. The current version of the OWL specification is OWL 2, standardised in 2009, which supersedes the earlier OWL 1 standard of 2004.

The building blocks of OWL are very similar to those of DLs, with the main difference that concepts are called *classes* and roles are called *properties*. Indeed, DLs have had a major influence on the development of OWL and its expressive features. Historically, however, OWL has also been conceived as an extension to RDF, a Web data modelling language whose expressivity is comparable to DL ABoxes. The formal semantics of RDF is subtly different from that of DLs, although both often lead to the same inferred consequences. Extending the RDF semantics to the expressive features of OWL improves compatibility between the two, but it also makes reasoning undecidable. Therefore, both styles of formal semantics were specified for OWL: the *Direct Semantics* based on DLs and the *RDF-based Semantics*.

The Direct Semantics of OWL is only defined for OWL ontologies which satisfy syntactic conditions that ensure that they can be read as $\mathcal{SROIQ}$ ontologies. This syntactic fragment of OWL is called *OWL DL*, while the unrestricted OWL language is called *OWL Full*. Large parts of OWL DL can be considered as a syntactic variant of $\mathcal{SROIQ}$. For example, the axiom $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$ would be written as follows in OWL:

```
EquivalentClasses( Mother ObjectIntersectionOf( Female Parent ) )
```

though one would use (abbreviations of) URIs instead of Mother, Female and Parent. The previous example uses the so-called *Functional-Style Syntax* of OWL, which is most directly related to DLs. Other syntactic forms are available, the RDF/XML serialisation being the most prominent.

There are still some differences between OWL DL under the Direct Semantics and $\mathcal{SROIQ}$. Syntactically, OWL provides many additional operators, such as special constructs for specifying domain and range of a property, which are logically redundant but convenient as shortcuts for common axioms.

Moreover, OWL also includes features that we excluded from our treatment of $\mathcal{SROIQ}$. The most important such feature are datatypes and datatype literals, which resemble classes and individual names with a pre-defined interpretation. For example, the Boolean datatype has exactly two elements—true and false—in any interpretation. Such pre-defined interpretation domains are called *concrete domains* in DL. Both DLs and OWL strictly distinguish roles/properties that relate to datatype values from those that relate to "abstract" individuals.

Besides the logical features, OWL also covers other aspects that are not considered in DLs. This includes means of naming ontologies and of importing ontological axioms from one ontology into another. Further extra-logical features include a simple form of *meta-modelling* called *punning*, non-logical axioms to *declare* identifiers, and support for adding *annotations* (comments) to arbitrary axioms and entities.

10 Further Reading

This article can only provide a first introduction to description logics and OWL. Further details and pointers to other introductory texts can be found in the Description Logic Primer [9]. For a more detailed coverage of OWL and its relationship to DL, we recommend the textbook *Foundations of Semantic Web Technologies* [6], which also treats the relationship of DLs to first-order logic. An in-depth treatment of DLs and related research topics is provided by the *Description Logic Handbook* [2], which also covers aspects of deduction algorithms and computational complexity.

A number of research papers focus on specific topics in DLs. Closely related to this article is the original article on $\mathcal{SROIQ}$, which also provides the details on structural restrictions that we omitted [7]. A detailed discussion of OWL datatypes and their description logic semantics is given by Motik and Horrocks [10]. Current developments in DL research are discussed at the annual DL Workshop (see http://dl.kr.org/ for proceedings) and at the major Semantic Web and Artificial Intelligence conferences.

The primary resources on OWL 2 are the online documents of the specification [11] where the OWL Primer provides a first introduction [5]. The differences of the 2009 OWL 2 standard to its predecessor are explained in [4].

Many related tools are available. The most popular free ontology editor is Protégé,[1] which can be used with a variety of OWL reasoners. Pointers to current OWL reasoners are best found online. Popular systems for large parts of OWL 2 DL ($\mathcal{SROIQ}$) include FaCT++, HermiT, Pellet and RacerPro. Some typical lightweight systems are

[1] http://protege.stanford.edu/

ELK (OWL EL), jCEL (OWL EL), Owlgress (OWL QL), OWLIM (OWL RL and QL), Quonto (OWL QL) and Snorocket (OWL EL). Details about these tools and related publications can be found on the respective homepages.

Acknowledgements We thank Fernando Bobillo, Peter Patel-Schneider, and Evgeny Zolin for helpful comments on an earlier version of this text. All authors have been working at the University of Oxford when writing this paper.

References

1. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope. In Leslie Pack Kaelbling and Alessandro Saffiotti, editors, *Proc. 19th Int. Joint Conf. on Artificial Intelligence (IJCAI'05)*, pages 364–369. Professional Book Center, 2005.
2. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, second edition, 2007.
3. Diego Calvanese, Guiseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
4. Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. OWL 2: The next step for OWL. *J. of Web Semantics*, 6:309–322, 2008.
5. Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph, editors. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-primer/.
6. Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.
7. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible $\mathcal{SROIQ}$. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *Proc. 10th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'06)*, pages 57–67. AAAI Press, 2006.
8. Markus Krötzsch. OWL 2 Profiles: An introduction to lightweight ontology languages. In Thomas Eiter and Thomas Krennwallner, editors, *Proceedings of the 8th Reasoning Web Summer School, Vienna, Austria, September 3–8 2012*, volume 7487 of *LNCS*, pages 112–183. Springer, 2012. Available at http://korrekt.org/page/OWL_2_Profiles.
9. Markus Krötzsch, František Simančík, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2012.
10. Boris Motik and Ian Horrocks. OWL datatypes: Design and implementation. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *Proc. 7th Int. Semantic Web Conf. (ISWC'08)*, volume 5318 of *LNCS*, pages 307–322. Springer, 2008.
11. W3C OWL Working Group. *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation, 27 October 2009. Available at http://www.w3.org/TR/owl2-overview/.