A Description Logic Primer*

Markus Krötzsch, František Simančík, Ian Horrocks
Department of Computer Science, University of Oxford, UK

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

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 Web Ontology Language OWL 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 they 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 intended 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 that reasoning algorithms of good performance are available. 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 paper we provide a self-contained first introduction to description logics. We start by explaining the basic way in which knowledge is modelled in DLs in Section 1 and continue with an intuitive introduction to the most important DL modelling features in Section 2. This leads us to the rather expressive DL called $\mathcal{SROIQ}$, the syntax of which we summarise in Section 3. In Section 4, we explain the underlying ideas of DL

*Version 1.0 of 19 January 2012. Comments and suggestions can be sent to Markus Krötzsch at markus.kroetzsch@cs.ox.ac.uk.

semantics and use it to define the meaning of $\mathcal{SROIQ}$ ontologies. Many DLs can be obtained by omitting some features of $\mathcal{SROIQ}$ and in Section 5 we review some of the most important DLs obtained in this way. In particular, this includes various light-weight description logics that allow for particularly efficient reasoning. In Section 6 we discuss the relationship of DLs to the Web Ontology Language OWL. We conclude with pointers to further reading in Section 7.

1. Basic Building Blocks of DL Ontologies

Description logics (DLs) provide means to model the relationships between entities in a domain of interest. In DLs there are three kinds of entities: concepts, roles and individual names.[1] Concepts denote sets of individuals, roles denote binary relations between the individuals, and individual names denote 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 the domain of people and their family relationships might use concepts such Parent to denote the set of all parents and Female to represent the set of all female individuals, roles such as parentOf to denote the (binary) relationship between parents and their children, and individual names such as julia and john to denote the individuals Julia and John.

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, each of which must be true in the situation described. These axioms typically capture only partial knowledge about the situation that the ontology is describing, and there may be many different states of the world that are consistent with the ontology. Although, from the point of view of logic, there is no principal difference between different types of axioms, it is customary to separate them into three groups: assertional (ABox) axioms, terminological (TBox) axioms and relational (RBox) axioms.

1.1. Asserting Facts with ABox Axioms

ABox axioms capture knowledge about named individuals, i.e., the concepts to which they belong and how they are related to each other. 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 denoted by parentOf to the individual named john.

[1] In OWL concepts and roles are respectively known as classes and properties; see Section 6.

The previous sentence shows that it can be rather cumbersome to explicitly point out that the relationships expressed by an axiom are really relationships between the individuals, sets and relations that are denoted by the respective individual names, concepts and roles. Assuming that this subtle distinction between syntactic identifiers and semantic entities is understood, we will thus often adopt a more sloppy and readable formulation. Section 4 below explains the underlying semantics with greater precision.

Although it is intuitively clear that Julia and John are different individuals, this fact does not logically follow from what we have stated so far. 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 actually different individuals. On the other hand, an *individual equality* assertion, such as

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

states that two different names are known to refer to the same individual. Such situations can arise, for example, when combining knowledge about the same domain from several different sources, a task that is known as *ontology alignment*.

1.2. 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 can be used 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, in practice one more often uses concept equivalence to give a name to complex expressions as introduced in Section 2.1 below. Furthermore, such additional concept expressions can be combined with equivalence and inclusion to describe more complex situations such as the disjointness of concepts, which asserts that two concepts do not share any instances.

1.3. Modelling Relationships between 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

$$\mathsf{parentOf} \sqsubseteq \mathsf{ancestorOf} \tag{7}$$

states that parentOf is a *subrole* of ancestorOf, i.e., every pair of individuals related by parentOf is also related by ancestorOf. Thus (2) and (7) 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

$$\mathsf{brotherOf} \circ \mathsf{parentOf} \sqsubseteq \mathsf{uncleOf}. \tag{8}$$

Note that role composition can only appear on the left-hand side of complex role inclusions. Furthermore, in order to retain decidability of reasoning (see the end of Section 4 for a discussion on decidability), their use is restricted by additional structural restrictions that specify whether or not 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. In DLs we can write *disjoint roles* as follows:

$$\mathit{Disjoint}(\mathsf{parentOf}, \mathsf{childOf}). \tag{9}$$

Further RBox axioms include *role characteristics* such as reflexivity, symmetry and transitivity of roles. These are closely related to a number of other DL features and we will discuss them again in more detail in Section 2.5.

2. Constructors for Concepts and Roles

The basic types of axioms introduced in Section 1 are rather limited for accurate modelling. To describe more complex situations, DLs allow new concepts and roles to be built using a variety of different constructors. We distinguish concept and role constructors depending on whether concept or role expressions are constructed. In the case of concepts, one can further separate basic Boolean constructors, role restrictions and nominals/enumerations. At the end of this section, we revisit the additional kinds of RBox axioms that have been omitted in Section 1.3.

2.1. Boolean Concept Constructors

Boolean concept constructors provide basic Boolean operations that are closely related to the familiar operations of intersection, union and complement of sets, or to conjunction, disjunction and negation of logical expressions.

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. DLs support such statements by allowing us to form complex concepts such as the *intersection* (also called *conjunction*)

$$\mathsf{Female} \sqcap \mathsf{Parent}, \tag{10}$$

which denotes the set of individuals that are both female and parents. A complex concept can be used in axioms in exactly the same way as an atomic concept, e.g., in the equivalence $\mathsf{Mother} \equiv \mathsf{Female} \sqcap \mathsf{Parent}$.

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

$$\mathsf{Father} \sqcup \mathsf{Mother} \tag{11}$$

describes those individuals that are either fathers or mothers. Again, it can be used in an axiom such as $\mathsf{Parent} \equiv \mathsf{Father} \sqcup \mathsf{Mother}$, which states that a parent is either 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. These could be described by the complex concept

$$\mathsf{Female} \sqcap \neg\mathsf{Married}, \tag{12}$$

where the *complement* (also called *negation*) $\neg\mathsf{Married}$ denotes the set of all individuals that are not married.

It is sometimes useful to be able 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

$$\top \sqsubseteq \mathsf{Male} \sqcup \mathsf{Female}, \tag{13}$$

where the *top concept* $\top$ is a special concept with every individual as an instance; it can be viewed as an abbreviation for $C \sqcup \neg C$ for an arbitrary concept C. Note that this modelling is rather coarse as it presupposes that every individual has a gender, which may not be reasonable for instances of a concept such as $\mathsf{Computer}$. We will see more useful applications for $\top$ later on.

To express that, for the purposes of our modelling, nobody can be both a male and a female at the same time, we can declare the set of male and the set of female individuals to be disjoint. While ontology languages like OWL provide a basic constructor for disjointness, it is naturally captured in DLs with the axiom

$$\mathsf{Male} \sqcap \mathsf{Female} \sqsubseteq \bot, \tag{14}$$

where the *bottom concept* $\bot$ is the dual of $\top$, that is the special concept with no individuals as instances; it can be seen as an abbreviation for $C \sqcap \neg C$ for an arbitrary concept C. The above axiom thus says that the intersection of the two concepts is empty.

2.2. Role Restrictions

So far we have seen how to use TBox and RBox axioms to express relationships between concepts and roles, respectively. The most interesting feature of DLs, however, 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 $\mathsf{parentOf}$, namely, a parent is someone who is a parent of at least one individual. In DLs, this relationship can be captured by the concept equivalence

$$\mathsf{Parent} \equiv \exists \mathsf{parentOf}.\top, \tag{15}$$

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 those individuals that are parents of at least one female individual, i.e., those that have a daughter.

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

$$\forall \mathsf{parentOf}.\mathsf{Female}. \tag{16}$$

It is a common error to forget that (16) also includes those that have no children at all. 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., no "no children that are not female." Following this wording, the concept (16) could indeed 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 are useful in combination with the top concept for expressing *domain* and *range restrictions* on roles; that is, restrictions on the kinds of individual that can be in the domain and range of a given role. To restrict the domain of sonOf to male individuals we can use the axiom

$$\exists \mathsf{sonOf}.\top \sqsubseteq \mathsf{Male}, \tag{17}$$

and to restrict its range to parents we can write

$$\top \sqsubseteq \forall \mathsf{sonOf}.\mathsf{Parent}. \tag{18}$$

In combination with the assertion $\mathsf{sonOf}(\mathsf{john}, \mathsf{julia})$, these axioms would then allow us to deduce that John is male and Julia is a parent. Note how this contrasts with the meaning of *constraints* in databases, which would also allow us to state, e.g., that all sons must be male. However, given only the fact that John is the son of Julia, such a constraint would simply be violated (leading to an error) rather than implying that John is male. Mistaking DL axioms for constraints is a very common source of modelling errors.

Number restrictions allow us to restrict the number of individuals that can be reached via a given role. For example, we can form the *at-least restriction*

$$\geqslant 2\ \mathsf{childOf}.\mathsf{Parent} \tag{19}$$

to describe the set of individuals that are children of at least two parents, and the *at-most restriction*

$$\leqslant 2\ \mathsf{childOf}.\mathsf{Parent} \tag{20}$$

for those that are children of at most two parents. 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 are talking to themselves is described by the concept

$$\exists \mathsf{talksTo}.\mathit{Self}. \tag{21}$$

2.3. Nominals

As well as defining concepts in terms of other concepts (and roles), it may also be useful to define a concept by simply enumerating its instances. For example, we might define the concept Beatle by enumerating its instances: john, paul, george, and ringo. Enumerations are not supported natively in DLs, but they can be simulated in DLs using *nominals*. A nominal is a concept that has exactly one instance. For example, {john} is the concept whose only instance is (the individual denoted by) john. Combining nominals with union, the enumeration in our example could be expressed as

$$\mathsf{Beatle} \equiv \{\mathsf{john}\} \sqcup \{\mathsf{paul}\} \sqcup \{\mathsf{george}\} \sqcup \{\mathsf{ringo}\}. \tag{22}$$

It is interesting to note that, using nominals, a concept assertion Mother(julia) can be turned into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \mathsf{Mother}$ and a role assertion parentOf(julia, john) into a concept inclusion $\{\mathsf{julia}\} \sqsubseteq \exists \mathsf{parentOf}.\{\mathsf{john}\}$. This illustrates that the distinction between ABox and TBox does not have a deeper logical meaning.

2.4. Role Constructors

In contrast to the variety of concept constructors, DLs provide only few constructor for forming complex roles. In practice, *inverse roles* are the most important such constructor. Intuitively, the relationship between the roles parentOf and childOf is that, for example, if Julia is a parent of John, then John is a child of Julia and vice versa. More formally, parenfOf is the inverse of childOf, which in DLs can be expressed by the equivalence

$$\mathsf{parentOf} \equiv \mathsf{childOf}^{-}, \tag{23}$$

where the complex role $\mathsf{childOf}^{-}$ denotes the inverse of childOf.

In analogy to the top concept, DLs also provide the *universal role*, denoted by U, which always relates all pairs of individuals. It typically plays a minor role in modelling,[2] but it establishes symmetry between roles and concepts w.r.t. a top element. Similarly, an *empty role* that corresponds to the bottom concept is also available in OWL but has rarely been introduced as a constructor in DLs; however, we can define any role R to be empty using the axiom $\top \sqsubseteq \neg\exists R.\top$ ("all things do not relate to anything through R"). Interestingly, the universal role cannot be defined by TBox axioms using the constructors introduced above, and in particular universal role restrictions cannot express that a role is universal.

2.5. More RBox Axioms: Role Characteristics

In Section 1.3 we introduced three forms of RBox axioms: role inclusions, role equivalences and role disjointness. OWL provides a variety of others, 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 ax-

[2] Although there are a few interesting things that could be expressed with U, such as *concept products* [12], tool support is rarely sufficient for using this feature in practice.

ioms are just syntactic sugar; all role characteristics can be expressed using the features of DLs that we have already introduced.

Transitivity is a special form of complex role inclusion. For example, transitivity of $\mathsf{ancestorOf}$ can be captured by the axiom $\mathsf{ancestorOf} \circ \mathsf{ancestorOf} \sqsubseteq \mathsf{ancestorOf}$. A role is *symmetric* if it is equivalent to its own inverse, e.g., $\mathsf{marriedTo} \equiv \mathsf{marriedTo}^-$, and it is *asymmetric* if it is disjoint from its own 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 $\top \sqsubseteq \neg\exists \mathsf{marriedTo}.\mathit{Self}$.

3. The Description Logic $\mathcal{SROIQ}$

In this section, we summarise the various features that have been introduced informally above to provide a comprehensive definition of DL syntax. Doing so yields the description logic called $\mathcal{SROIQ}$, which is one of the most expressive DLs commonly considered today. It also largely agrees in expressivity with the ontology language OWL 2 DL, though there are still some differences as explained in Section 6.

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. Now the set of $\mathcal{SROIQ}$ *role expressions* $\mathbf{R}$ (over this signature) is defined by the following grammar:

$$\mathbf{R} ::= U \mid \mathsf{N}_R \mid \mathsf{N}_R{}^-$$

where U is the universal role (Section 2.4). Based on this, the set of $\mathcal{SROIQ}$ *concept expressions* $\mathbf{C}$ is defined as:

$$\mathbf{C} ::= \mathsf{N}_C \mid (\mathbf{C} \sqcap \mathbf{C}) \mid (\mathbf{C} \sqcup \mathbf{C}) \mid \neg\mathbf{C} \mid \top \mid \bot \mid \exists \mathbf{R}.\mathbf{C} \mid \forall \mathbf{R}.\mathbf{C} \mid \geqslant n\, \mathbf{R}.\mathbf{C} \mid \leqslant n\, \mathbf{R}.\mathbf{C} \mid \exists \mathbf{R}.\mathit{Self} \mid \{\mathsf{N}_I\}$$

where n is a non-negative integer. As usual, expressions like $(\mathbf{C} \sqcap \mathbf{C})$ represent any expression of the form $(C \sqcap D)$ with $C, D \in \mathbf{C}$. It is common to omit parentheses if this cannot lead to confusion with expressions of different semantics. For example, parentheses do not matter for $A \sqcup B \sqcup C$ whereas the expressions $A \sqcap B \sqcup C$ and $\exists R.A \sqcap B$ are ambiguous.

Using the above sets of individual names, roles and concepts, the *axioms* of $\mathcal{SROIQ}$ can be defined to be of the following basic forms:

ABox:	$\mathbf{C}(\mathsf{N}_I)$	$\mathbf{R}(\mathsf{N}_I, \mathsf{N}_I)$	$\mathsf{N}_I \approx \mathsf{N}_I$	$\mathsf{N}_I \not\approx \mathsf{N}_I$
TBox:	$\mathbf{C} \sqsubseteq \mathbf{C}$	$\mathbf{C} \equiv \mathbf{C}$		
RBox:	$\mathbf{R} \sqsubseteq \mathbf{R}$	$\mathbf{R} \equiv \mathbf{R}$	$\mathbf{R} \circ \mathbf{R} \sqsubseteq \mathbf{R}$	$\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$

with the intuitive meanings as explained in Section 1 and 2.

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 on ontologies. These restrictions refer not to single axioms but to the structure of the ontology as a whole, hence they are called *structural restrictions*. The two concrete such conditions relevant for $\mathcal{SROIQ}$ are based on the notions of *simplicity* and *regularity*. Notably, both are automatically satisfied for ontologies that do not contain complex role inclusion axioms.

A role R in an ontology $\mathcal{O}$ is called *non-simple* if some complex role inclusion axiom (i.e., one that uses role composition $\circ$) in $\mathcal{O}$ implies instances of R; otherwise it is called *simple*. To be more precise, we first define the *subroles* of a role R as follows:

- R is a subrole of itself,
- if R' is a subrole of R and $\mathcal{O}$ contains an axiom $T \sqsubseteq R'$, $T \equiv R'$ or $R' \equiv T$, then T is a subrole of R.

Now the role R is non-simple if the ontology contains an axiom $S \circ T \sqsubseteq R'$ where R' is a subrole of R. All other roles are called simple.[3] Now for a $\mathcal{SROIQ}$ ontology it is required that the following axioms and concepts contain simple roles only:

Restricted axioms:	$\mathit{Disjoint}(\mathbf{R}, \mathbf{R})$
Restricted concept expressions:	$\exists \mathbf{R}.\mathit{Self} \qquad \geqslant n\, \mathbf{R}.\mathbf{C} \qquad \leqslant n\, \mathbf{R}.\mathbf{C}.$

The other structural restriction that is relevant for $\mathcal{SROIQ}$ is called *regularity* and is concerned with RBox axioms only. Roughly speaking, the restriction ensures that cyclic dependencies between complex role inclusion axioms occur only in a limited form. For details, please see the pointers given in Section 7. For the introductory treatment in this paper, it suffices to note that regularity, just like simplicity, is a property of the ontology as a whole that cannot be checked for each axiom individually. An important practical consequence is that the union of two regular ontologies may no longer be regular. This must be taken into account when merging ontologies in practice.

4. Description Logic Semantics

The formal meaning of DL axioms is given by their 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 description logics is not difficult to understand provided that some common misconceptions are avoided.

Intuitively speaking, an ontology describes a particular situation in a given domain of discourse. For example, the axioms in Sections 1 and 2 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

[3] Whether the universal role U is simple or not is a matter of preference that does not affect the computational properties of the logic [13]. However, the universal role in OWL 2 is considered non-simple.

intended meaning of the identifiers in our ontologies has no influence on their formal semantics: what we know about them stems only from the ontological axioms. On the other hand, the axioms in an ontology typically do not provide complete information. For example, (3) and (4) in Section 1.1 state that some individuals are equal and that others are unequal, but in many other cases this information might be left unspecified.

Description logics have been designed to deal with 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 characteristic is sometimes called the *Open World Assumption* since it keeps unspecified information open.[4] 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 what is said in the ontology. The more axioms an ontology contains, the more specific are the constraints that it imposes on possible interpretations, and the fewer interpretations exist that satisfy all of the axioms. Conversely, if fewer interpretations satisfy an ontology, then more axioms hold in all of them, and more logical consequences follow from the ontology. The previous two sentences imply that the semantics of description logics 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*. In this case *every* axiom holds vacuously in all of the (zero) interpretations that satisfy the ontology. Such an ontology is clearly of no utility, and avoiding inconsistency (and checking for it in the first place) is therefore an important task during modelling.

We have outlined above the most important ideas of DL semantics. What remains to be done is to define what we really mean by an "interpretation" and which conditions must hold for particular axioms to be satisfied by an interpretation. For this, we closely follow the intuitive ideas established above: an interpretation $\mathcal{I}$ consists of a set $\Delta^{\mathcal{I}}$ called the *domain* of $\mathcal{I}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps each atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, each atomic role R to a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$, and each individual name a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The interpretation of complex concepts and roles follows from the interpretation of the basic entities. Table 1 shows how to obtain the semantics of each compound expression from the semantics of its parts. By "$R^{\mathcal{I}}$-successor of x" we mean any individual y such that $\langle x, y\rangle \in R^{\mathcal{I}}$. The definition should confirm the intuitive explanations given for each case in Section 2. For example, the semantics of $\mathsf{Female} \sqcap \mathsf{Parent}$ is indeed the intersection of the semantics of Female and Parent.

Since an interpretation $\mathcal{I}$ fixes the meaning of all entities, we can unambiguously say for each axiom whether it holds in $\mathcal{I}$ or not. An axiom *holds* in $\mathcal{I}$ (we also say $\mathcal{I}$ *satisfies* α and write $\mathcal{I} \models \alpha$) if the corresponding condition in Table 2 is met. Again, these definitions fully agree with the intuitive explanations given in Section 1. If all axioms in an ontology $\mathcal{O}$ hold in $\mathcal{I}$ (i.e., if $\mathcal{I}$ satisfies $\mathcal{O}$, written $\mathcal{I} \models \mathcal{O}$), then $\mathcal{I}$ is a *model* of $\mathcal{O}$. Thus a model is an abstraction of a state of the world that satisfies all axioms in the ontology. An ontology is *consistent* if it has at least one model. An axiom α is a

[4] A *Closed World Assumption* "closes" the interpretation by assuming that every fact not explicitly stated to be true is actually false. Both terms are not formally specified and rather outline the general flavour of a semantics than any particular definition.

Table 1. Syntax and semantics of $\mathcal{SROIQ}$ constructors

	Syntax	Semantics
Individuals:		
individual name	a	$a^{\mathcal{I}}$
Roles:		
atomic role	R	$R^{\mathcal{I}}$
inverse role	R^{-}	$\{\langle x, y\rangle \mid \langle y, x\rangle \in R^{\mathcal{I}}\}$
universal role	U	$\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Concepts:		
atomic concept	A	$A^{\mathcal{I}}$
intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
complement	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
top concept	$\top$	$\Delta^{\mathcal{I}}$
bottom concept	$\bot$	$\emptyset$
existential restriction	$\exists R.C$	$\{x \mid \text{some } R^{\mathcal{I}}\text{-successor of } x \text{ is in } C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{x \mid \text{all } R^{\mathcal{I}}\text{-successors of } x \text{ are in } C^{\mathcal{I}}\}$
at-least restriction	$\geqslant n\, R.C$	$\{x \mid \text{at least } n\ R^{\mathcal{I}}\text{-successors of } x \text{ are in } C^{\mathcal{I}}\}$
at-most restriction	$\leqslant n\, R.C$	$\{x \mid \text{at most } n\ R^{\mathcal{I}}\text{-successors of } x \text{ are in } C^{\mathcal{I}}\}$
local reflexivity	$\exists R.Self$	$\{x \mid \langle x, x\rangle \in R^{\mathcal{I}}\}$
nominal	$\{a\}$	$\{a^{\mathcal{I}}\}$

where $a, b \in \mathsf{N}_I$ are individual names, $A \in \mathsf{N}_C$ is a concept name, $C, D \in \mathbf{C}$ are concepts, $R \in \mathbf{R}$ is a role

Table 2. Syntax and semantics of $\mathcal{SROIQ}$ axioms

	Syntax	Semantics
ABox:		
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a, b)$	$\langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}}$
individual equality	$a \approx b$	$a^{\mathcal{I}} = b^{\mathcal{I}}$
individual inequality	$a \not\approx b$	$a^{\mathcal{I}} \neq b^{\mathcal{I}}$
TBox:		
concept inclusion	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
concept equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
RBox:		
role inclusion	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role equivalence	$R \equiv S$	$R^{\mathcal{I}} = S^{\mathcal{I}}$
complex role inclusion	$R_1 \circ R_2 \sqsubseteq S$	$R_1^{\mathcal{I}} \circ R_2^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
role disjointness	$Disjoint(R, S)$	$R^{\mathcal{I}} \cap S^{\mathcal{I}} = \emptyset$

consequence of an ontology $\mathcal{O}$ (or $\mathcal{O}$ *entails* α written $\mathcal{O} \models \alpha$) if α holds in every model of $\mathcal{O}$. In particular, an inconsistent ontology entails every axiom.

A noteworthy consequence of this semantics is 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). Possibly even more important is the fact that the domain of an interpretation is allowed to contain many individuals

that are not denoted by any individual name. A common confusion in modelling arises from the implicit assumption that interpretations must only contain individuals that are denoted 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 3 children when only one (John) is given. However, there are many conceivable models where Julia does have three children, even though only one of them is explicitly named. A significant number of modelling errors can be traced back to similar misconceptions that are easy to prevent if the general open world assumption of DLs is kept in mind.

Another point to note is that the above specification of the semantics does not provide any hint as to how to compute the relevant entailments in practical software tools. There are infinitely many possible interpretations, each of which may have an infinite domain (in fact there are some ontologies that are satisfied only by interpretations with infinite domains). 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 concrete 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). For our purposes it suffices to know that entailment of axioms is decidable for $\mathcal{SROIQ}$ (with the structural restrictions explained in Section 3) and that a number of free and commercial tools are available. Such tools are typically optimised for more specific reasoning problems, such as consistency checking, the entailment of concept subsumptions (subsumption checking) or of concept assertions (instance checking). Many of these standard inferencing problems can be expressed in terms of each other, so they can be handled by very similar reasoning algorithms.

5. Important Fragments of $\mathcal{SROIQ}$

Many different description logics have been introduced in the literature. Typically, they can be characterised by the types of constructors and axioms that they allow, which are often a subset of the constructors in $\mathcal{SROIQ}$. For example, the description logic $\mathcal{ALC}$ is the fragment of $\mathcal{SROIQ}$ that allows no RBox axioms and only $\sqcap$, $\sqcup$, $\neg$, $\exists$ and $\forall$ as its concept constructors. It is often considered the most basic DL. The extension of $\mathcal{ALC}$ with transitive roles is traditionally denoted by the letter $\mathcal{S}$. Some other letters used in DL names hint at a particular constructor, such as inverse roles $\mathcal{I}$, nominals $\mathcal{O}$, qualified number restrictions $\mathcal{Q}$, and role hierarchies (role inclusion axioms without composition) $\mathcal{H}$. So, for example, the DL named $\mathcal{ALCHIQ}$ extends $\mathcal{ALC}$ with role hierarchies, inverse roles and qualified number restrictions. The letter $\mathcal{R}$ most commonly refers to the presence of role inclusions, local reflexivity *Self*, and the universal role U, as well as the additional role characteristics of transitivity, symmetry, asymmetry, role disjointness, reflexivity, and irreflexivity. This naming scheme explains the name $\mathcal{SROIQ}$.

In recent years, fragments of DLs have been specifically developed in order to obtain favourable computational properties. For this purpose, $\mathcal{ALC}$ is already too large,

since it only admits reasoning algorithms that run in worst-case exponential time. More light-weight DLs can be obtained by further restricting expressivity, while at the same time a number of additional $\mathcal{SROIQ}$ features can be added without loosing the good computational properties. The three main approaches for obtaining light-weight DLs are $\mathcal{EL}$, *DLP* and *DL-Lite*, which also correspond to language fragments OWL EL, OWL RL and OWL QL of the Web Ontology Language.

The $\mathcal{EL}$ family of description logics is characterised by allowing unlimited use of existential quantifiers and concept intersection. The original description logic $\mathcal{EL}$ allows only those features and $\top$ but no unions, complements or universal quantifiers, and no RBox axioms. Further extensions of this language are known as $\mathcal{EL}^+$ and $\mathcal{EL}^{++}$. The largest such extension allows the constructors $\sqcap$, $\top$, $\bot$, $\exists$, *Self*, nominals and the universal role, and it supports all types of axioms other than role symmetry, asymmetry and irreflexivity. Interestingly, all standard reasoning tasks for this DL can still be solved in worst-case polynomial time. One can even drop the structural restriction of regularity that is important for $\mathcal{SROIQ}$. $\mathcal{EL}$-type ontologies have been used to model large but light-weight ontologies that consist mainly of terminological data, in particular in the life sciences. A number of reasoners are specifically optimised for handling $\mathcal{EL}$-type ontologies, the most recent of which is the ELK reasoner for OWL EL.[5]

DLP is short for *Description Logic Programs* and comprises various DLs that are syntactically restricted in such a way that axioms could also be read as rules in first-order Horn logic without function symbols. Due to this, DLP-type logics can be considered as kinds of rule languages (hence the name OWL RL) contained in DLs. To accomplish this, one has to allow different syntactic forms for subconcepts and superconcepts in concept inclusion axioms. We do not provide the details here. While DLs in general may require us to consider domain elements that are not denoted by individual names, for DLP one can always restrict attention to models in which all domain elements are denoted by individual names. 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 the Oracle 11g database management system.

DL-Lite is a family of DLs that is also used in combination with large data collections and existing databases, in particular to augment the expressivity of a query language that retrieves such data. This approach, known as Ontology Based Data Access, considers ontologies as a language for constructing *views* or *mapping rules* on top of existing data. The core feature of DL-Lite is that data access can be realised with standard query languages such as SQL that are not aware of the DL semantics. Ontological information is merely used in a query preprocessing step. Like DLP, DL-Lite requires different syntactic restrictions for subconcepts and superconcepts. We do not present the details here.

6. Relationship to OWL

The *Web Ontology Language* OWL is a knowledge representation language standardised by the World Wide Web Consortium (W3C). OWL is one of the most important applications of description logics today. In this section, we briefly outline the relationship of the two languages. A comprehensive treatment is beyond the scope of this paper; see

[5] http://elk-reasoner.googlecode.com/

Section 7 for pointers to further reading. The current version of the OWL specification is OWL 2 as standardised in 2009. This supersedes the earlier OWL 1 standard of 2004.

The main building blocks of OWL are indeed very similar to those of DLs, with the main difference that concepts are called *classes* and roles are called *properties*. It is therefore not surprising that description logics have had a major influence on the development of OWL and the expressive features that it provides. 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, even though both lead to the same consequences in many common cases. Extending the RDF semantics to the expressive features of OWL improves the compatibility between the two, but it also makes reasoning undecidable. Therefore, it has been decided to specify both styles of formal semantics for OWL: the *Direct Semantics* based on DLs and the *RDF-based Semantics*.

In this section, we are therefore mainly interested in the Direct Semantics of OWL. This semantics is only defined for OWL ontologies that abide by certain syntactic restrictions (essentially the restriction that the OWL axioms can be read as $\mathcal{SROIQ}$ axioms for which the structural restrictions of Section 3 are satisfied). This syntactic fragment of OWL is called *OWL DL*.[6] Under the Direct Semantics, large parts of OWL DL can indeed 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 ) )
```

where the symbols Mother, Female and Parent would be identifier strings that conform to the OWL specification.[7] The above example illustrates the close relationship between the syntax of $\mathcal{SROIQ}$ and that of OWL. In many cases, it is indeed enough to translate an operator symbol of $\mathcal{SROIQ}$ into the corresponding operator name in OWL, which is then written in prefix notation like a function. This is also why the above form of syntax is called *Functional-Style Syntax*. The OWL standard provides a number of syntactic forms that can be used to express OWL ontologies. The most prominent among these is the RDF/XML serialisation since it is the only format that all conforming OWL tools need to understand. On the other hand, it is more difficult for humans to read and we do not present it here.

It is interesting to note that there are still a few differences between OWL DL under the Direct Semantics and $\mathcal{SROIQ}$. On a syntactic level, OWL provides a lot more operators that, though logically redundant, can be convenient as shortcuts for compound DL axioms. For example, OWL has special constructs for specifying domain and range of a property, even though these could equally well be expressed as in Section 2.2. These kinds of features also include the empty (bottom) property, which can easily be defined but is not included as a language feature in DLs.

However, OWL also includes some expressive features that we did not include in our treatment of $\mathcal{SROIQ}$ above. Most notably, this includes support for datatypes and datatype literals. These behave like classes and individual names but come with a fixed,

[6] In contrast, the OWL language without any syntactic constraints is called *OWL Full*. It comprises ontologies that can only be interpreted under the RDF-based Semantics.

[7] Entity names in OWL are generally based on Uniform Resource Identifiers (URIs). The details are not relevant here.

pre-defined interpretation. For example, the datatype for Boolean values has exactly two elements – true and false – in any interpretation. This can also be introduced in DLs by so-called *concrete domains*, i.e., pre-defined interpretation domains. Both DLs and OWL in this case strictly distinguish roles/properties that relate to "abstract" individuals from those that relate to values from some datatype. In OWL, the constructs that relate to datatypes include "Data" in their name while constructs that relate to abstract individuals include "Object." For example, OWL distinguishes `ObjectIntersectionOf` (used above) from `DataIntersectionOf` (the intersection of datatypes).

The only other logical feature that is missing in DLs are so-called *Keys*. These are special forms of rules that can be used for data integration. Roughly speaking, a key specifies that two named individuals are entailed to be equal if they agree on certain property values and class memberships, similar to key constraints in databases. For example, the combination of nationality and registration number might be treated as a key for (i.e., sufficient to uniquely identify) motor vehicles.

Besides the logical features, OWL also includes a number of other aspects that are not considered in description logics at all. For example, it includes means of naming an ontology 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 the possibility to add *annotations* to arbitrary axioms and entities similar to comments in a programming language.

7. Further Reading

This paper can only provide a first introduction to description logics and OWL. Further details, especially regarding formal semantics and modelling, can be found in the extensive lecture notes for the course *Foundations of Description Logics*, given at the *Reasoning Web Summer School 2011* [11]. For a more detailed coverage of OWL and its relationship to DL, we recommend the textbook *Foundations of Semantic Web Technologies* [7]. This introductory text also treats the relationship of DLs to first-order logic, DL query answering and extensions for rule-based modelling (related to keys in OWL), which we have omitted here. An in-depth treatment of description logics and related research topics is provided by the *Description Logic Handbook*, which also covers interesting aspects of deduction algorithms and computational complexity that are beyond the scope of this paper [2].

A number of research papers focus on specific topics in DLs. Closely related to this paper is the original article on $\mathcal{SROIQ}$ that also provides the details on regularity conditions that have been skipped above [8]. There are also various works that focus on $\mathcal{EL}$ [1,9], DLP [5] and DL-Lite [3]. 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 resource on OWL 2 are the online documents of the specification [10] where the OWL Primer provides a first introduction [6]. The differences of the 2009 OWL 2 standard to its predecessor are explained in [4].

Many related tools such as reasoners and ontology editors are available. The most popular free ontology editor is Protégé,[8] which can be used with a variety of OWL rea-

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

soners. Pointers to current OWL reasoners are best found online.[9] Popular systems for large parts of OWL 2 DL ($\mathcal{SROIQ}$) include FaCT++, HermiT, Pellet and RacerPro. Some typical light-weight systems are 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.

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] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: combining logic programs with description logic. In *Proc. 12th Int. Conf. on World Wide Web (WWW'03)*, pages 48–57. ACM, 2003.

[6] 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/.

[7] Pascal Hitzler, Markus Krötzsch, and Sebastian Rudolph. *Foundations of Semantic Web Technologies*. Chapman & Hall/CRC, 2009.

[8] 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.

[9] Markus Krötzsch. Efficient rule-based inferencing for OWL EL. In Toby Walsh, editor, *Proc. 22nd Int. Conf. on Artificial Intelligence (IJCAI'11)*, pages 2668–2673. AAAI Press/IJCAI, 2011.

[10] 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/.

[11] Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter F. Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data – 7th International Summer School 2011*, volume 6848 of *LNCS*, pages 76–136. Springer, 2011.

[12] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. All elephants are bigger than all mice. In Franz Baader, Carsten Lutz, and Boris Motik, editors, *Proc. 21st Int. Workshop on Description Logics (DL'08)*, volume 353 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[13] Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Cheap Boolean role constructors for description logics. In Steffen Hölldobler, Carsten Lutz, and Heinrich Wansing, editors, *Proc. 11th European Conf. on Logics in Artificial Intelligence (JELIA'08)*, volume 5293 of *LNAI*, pages 362–374. Springer, 2008.

[9] A list of reasoners can be found, e.g., at http://semanticweb.org/wiki/Category:Reasoner.