3.1: Description Logics Primer

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

The capability of inferring additional knowledge increases the modeling power of DLs but it also requires some understanding on the side of the modeler 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.

We provide a self-contained first introduction to Description Logics. We start by explaining the basic way in which knowledge is modeled in DLs and continue with an intuitive introduction to the most important DL modeling features. We explain the underlying ideas of DL semantics. Then a common starting point of DLs is introduced, being \( \mathcal{ALC} \), and subsequently the rather expressive DL called \( \mathcal{SROIQ} \), which forms the basis of the OWL 2 DL language. Many DLs can be obtained by omitting some features of \( \mathcal{SROIQ} \) and 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 and are also standardized, as fragments of OWL.

### 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. Concepts denote sets of individuals, roles denote sets of binary relations between the individuals\(^2\), and individual names denote single individuals in the domain. Readers familiar with first-order logic will recognize these as unary predicates, binary predicates and constants.

For example, an ontology representing the domain of people and their family relationships might use concepts such \texttt{Parent} to denote the set of all parents and \texttt{Female} to represent the set of all female individuals, roles such as \texttt{parentOf} to denote the (binary) relationship between parents and their children, and individual names such as \texttt{julia} and \texttt{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.

### 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

\[
\texttt{Mother(Julia)}
\]

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

Role assertions describe relations between named individuals. The assertion

\[
\texttt{parentOf(julia,john)}
\]

for example, states that Julia is a parent of John or, more precisely, that the individual named \texttt{julia} is in the relation that is denoted by \texttt{parentOf} to the individual named \texttt{john}. 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. Below the underlying semantics with greater precision\(^3\) is explained.
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 may or may not make the unique name assumption, so different names might refer to the same individual unless explicitly stated otherwise\(^4\). The individual inequality assertion

\[
\texttt{Julia} \not\approx \texttt{John}
\]

is used to assert that Julia and John are actually different individuals. On the other hand, an individual equality assertion, such as

\[
\texttt{John} \approx \texttt{Johnny}
\]

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.

**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

\[
\texttt{Mother} \sqsubseteq \texttt{Parent}
\]

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, (3.1.1) and (3.15) together imply that Julia is a parent.

*Concept equivalence* asserts that two concepts have the same instances, as in

\[
\texttt{Person} \equiv \texttt{Human}
\]

While synonyms may be perceived to be an obvious example of equivalent concepts, in practice one uses concept equivalence to give a name to complex expressions as introduced below and put real synonyms in the annotations or extra labels in the ontology file. 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.

**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

\[
\texttt{parentOf} \sqsubseteq \texttt{ancestorOf}
\]

states that parentOf is a subrole of ancestorOf, i.e., every pair of individuals related by parentOf is also related by ancestorOf. Thus (3.1.2) and (3.1.7) 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

\[
\texttt{brotherOf} \circ \texttt{parentOf} \subseteq \texttt{uncleOf}
\]

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 Appendix C for a recap on complexity and 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:

\[
\text{Disjoint}(\texttt{parentOf}, \texttt{childOf})
\]

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.

---

**Constructors for Concepts and Roles**

The basic types of axioms introduced earlier are rather limited for accurate modeling. 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 earlier.

**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*)

\[
\texttt{Female} \cap \texttt{Parent}
\]

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 \(\texttt{Mother} \equiv \texttt{Female} \cap \texttt{Parent} \).)

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

\[
\texttt{Father} \cup \texttt{Mother}
\]
describes those individuals that are either (biological) fathers or mothers. Again, it can be used in an axiom such as \\
(\texttt{Parent} \equiv \texttt{Father} \cup \texttt{Mother}), which states that a parent is either a father or a mother (and vice versa). One can extend this further, of course, by stating, e.g., that a parent is equivalent to father or mother or guardian: \\
(\texttt{Parent} \equiv \texttt{Father} \sqcup \texttt{Mother} \sqcup \texttt{Guardian}).

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
\\n(\texttt{Female} \sqcap \neg \texttt{Married})

where the complement (also called negation) \(\neg \texttt{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
\\n(\top \sqsubseteq \texttt{Male} \cup \texttt{Female})

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 modeling is rather coarse as it presupposes that every individual has a gender, which may not be reasonable, especially for instances of a concept such as Computer. We will see more useful applications for \(\top\\) later on.

To express that, for the purposes of our modeling, nobody can be both a parent and childless at the same time, we can declare the set of parents and the set of childless individuals to be disjoint. While ontology languages like OWL provide a basic constructor for disjointness, it is naturally captured in DLs with the axiom
\\n(\texttt{Parent} \sqcap \texttt{Childless} \sqsubseteq \bot)

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.

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 \texttt{Parent} and the role \texttt{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
\\n(\texttt{Parent} \equiv \exists \texttt{parentOf.} \top)

where the existential restriction \(\exists \texttt{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 \(\neg \exists \texttt{parentOf.} \top\\)
texttt{parentOf.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

\[\forall\texttt{parentOf.Female}\]

It is a common error to forget that (3.1.16) 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 (3.1.16) could indeed be equivalently expressed as \[\neg\exists\texttt{parentOf.}\neg\texttt{Female}\] (recall the equivalences in 2.1: First Order Logic Syntax and Semantics). 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\texttt{parentOf.}\top\sqcap\forall\texttt{parentOf.Female}\)]. To state the difference between Eq. 3.1.16 and the latter in another way: Eq. 3.1.16 says “if has children, then all female” and latter “does have children, and all 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 \texttt{sonOf} to male individuals we can use the axiom

\[\exists\texttt{sonOf.}\top\sqsubseteq\texttt{Male}\]

and to restrict its range to parents we can write

\[\top\sqsubseteq\forall\texttt{sonOf.Parent}\]

In combination with the assertion \texttt{sonOf(john, julia)}, these axioms would then allow us to deduce that John is male and Julia is a parent.

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

\[\geq\texttt{2childOf.Parent}\]

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

\[\leq\texttt{2childOf.Parent}\]

for those that are children of at most two parents. The axiom \[\texttt{Person}\sqsubseteq\geq\texttt{2childOf.Parent}\] then states that every person is a child of exactly two parents. A shorthand notation is \[\texttt{Person}\sqsubseteq=\texttt{2childOf.Parent}\]

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

\[\exists\texttt{talksTo. Self}\]
**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

\[
\text{Beatle} \equiv \{\text{john}\} \sqcup \{\text{paul}\} \sqcup \{\text{george}\} \sqcup \{\text{ringo}\}
\]

It is interesting to note that, using nominals, a concept assertion Mother(julia) can be turned into a concept inclusion \(\{\text{julia}\} \sqsubseteq \text{Mother}\) and a role assertion parentOf(julia, john) into a concept inclusion \(\{\text{julia}\} \sqsubseteq \exists \text{parentOf.}\{\text{john}\}\). This illustrates that the distinction between ABox and TBox does not have a deeper logical meaning\(^5\).

**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, parentOf is the inverse of childOf, which in DLs can be expressed by the equivalence

\[
\text{parentOf} \equiv \text{childOf}^{-}
\]

where the complex role childOf\(^{-}\) denotes the inverse of childOf.

In analogy to the top concept, DLs also provide the *universal role*, denoted by \(\langle U \rangle\), which always relates all pairs of individuals. It typically plays a minor role in modelling,\(^6\) 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 \(\langle R \rangle\) to be empty using the axiom \(\top \sqsubseteq \neg \exists R.\top\) (“all things do not relate to anything through \(\langle R \rangle\)”). 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.

**More RBox Axioms: Role Characteristics**

In "Basic Building Blocks of DL Ontologies" 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 \(\langle \text{Trans(}\langle \text{ancestorOf}\rangle)\rangle\) to express that the role ancestorOf is transitive. However, such axioms 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 `ancestorOf` can be captured by the axiom `	exttt{ancestorOf}\circ\texttt{ancestorOf}\sqsubseteq\texttt{ancestorOf}`. A role is symmetric if it is equivalent to its own inverse, e.g., `	exttt{marriedTo}\equiv\texttt{marriedTo}^{-}`), and it is asymmetric if it is disjoint from its own inverse, as in `	exttt{Disjoint(}\texttt{parentOf, parentOf}^{-}\texttt{)}\texttt{)}`. If desired, global reflexivity can be expressed by imposing local reflexivity on the top concept as in `	exttt{top}\sqsubseteq\exists\texttt{knows}.\texttt{Self}`. A role is irreflexive if it is never locally reflexive, as in the case of `	exttt{top}\sqsubseteq\neg\exists\texttt{marriedTo}.\texttt{Self}`.

### 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 modeling 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 "Basic Building Blocks of DL Ontologies" and "Constructors for Concepts and Roles" 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 our ontologies has no influence on their formal semantics: what we know about them stems only from the axioms. On the other hand, the axioms in an ontology typically do not provide complete information. For example, Eqs. (3.1.3) and (3.1.4) in "Basic Building Blocks of DL Ontologies" 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 called the Open World Assumption since it keeps unspecified information open.\(^7\)

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Individuals:</strong></td>
<td></td>
</tr>
<tr>
<td>individual name</td>
<td>(\texttt{\langle a\rangle})</td>
</tr>
<tr>
<td>atomic role</td>
<td>(\texttt{\langle R\rangle})</td>
</tr>
<tr>
<td><strong>Roles:</strong></td>
<td></td>
</tr>
<tr>
<td>inverse role</td>
<td>(\texttt{\langle R^{-}\rangle})</td>
</tr>
<tr>
<td>universal role</td>
<td>(\texttt{\langle U\rangle})</td>
</tr>
<tr>
<td><strong>Concepts:</strong></td>
<td></td>
</tr>
<tr>
<td>atomic concept</td>
<td>(\texttt{\langle A\rangle})</td>
</tr>
<tr>
<td>Syntax</td>
<td>Semantics</td>
</tr>
<tr>
<td>------------</td>
<td>-----------</td>
</tr>
<tr>
<td>intersection</td>
<td>( \cap )</td>
</tr>
<tr>
<td>union</td>
<td>( \cup )</td>
</tr>
<tr>
<td>complement</td>
<td>( \neg )</td>
</tr>
<tr>
<td>top concept</td>
<td>( \top )</td>
</tr>
<tr>
<td>bottom concept</td>
<td>( \bot )</td>
</tr>
<tr>
<td>existential restriction</td>
<td>( \exists )</td>
</tr>
<tr>
<td>universal restriction</td>
<td>( \forall )</td>
</tr>
<tr>
<td>at-least restriction</td>
<td>( \geqslant n )</td>
</tr>
<tr>
<td>at-most restriction</td>
<td>( \leqslant n )</td>
</tr>
<tr>
<td>local reflexivity</td>
<td>( \exists )</td>
</tr>
<tr>
<td>nominal</td>
<td>( { a } )</td>
</tr>
</tbody>
</table>

where \( a, b \in \texttt{N}_I \) are individual names, \( A \in \texttt{N}_C \) is a concept name, \( C, D \in \textbf{C} \) are concepts, and \( R \in \textbf{R} \) is a role.

Table 3.1.1: Syntax and semantics of \( \text{mathcal}(\text{SROIQ}) \) constructors.

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 (recall "1.2: What is an Ontology?" with high precision). 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 ontology development.

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 \(\{A\}\subseteq\Delta^\mathcal{I}\), each atomic role \(\{R\}\) to a binary relation \(\{R\}\subseteq\Delta^\mathcal{I}\times\Delta^\mathcal{I}\), and each individual name \(\{a\}\) to an element \(\{a\}\in\Delta^\mathcal{I}\). The interpretation of complex concepts and roles follows from the interpretation of the basic entities. Table 3.1.1 shows how to obtain the semantics of each compound expression from the semantics of its parts. By \(\{R\}\)-successor of \(\{x\}\) we mean any individual \(\{y\}\) such that \(\{x\}, y\in\{R\}\). For each atom of a formula \(\{\mathcal{A}\}\), there is a corresponding boolean expression \(\mathcal{A}\) that can be evaluated for any interpretation \(\mathcal{I}\). Table 3.1.2 shows how to obtain the semantics of each compound expression from the semantics of its parts. By “\(\{R\}\)-successor of \(\{x\}\)” we mean any individual \(\{y\}\) such that \(\{x\}, y\in\{R\}\).
The definition should confirm the intuitive explanations given for each case in "Constructors for Concepts of Roles". For example, the semantics of \((\text{Female} \sqcap \text{Parent})\) 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 \(\alpha\) and write \(\mathcal{I} \models \alpha\)) if the corresponding condition in Table 3.1.2 is met. Again, these definitions fully agree with the intuitive explanations given in "Basic Building Blocks of DL Ontologies". 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 \(\alpha\) is a consequence of an ontology \(\mathcal{O}\) (or \(\mathcal{O}\) entails \(\alpha\) written \(\mathcal{O} \models \alpha\)) if \(\alpha\) 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

\[
\text{parentOf(julia,john)} \\
\text{manyChildren(julia)} \\
\text{manyChildren}\sqsubseteq\text{geqslant3parentOf}\top
\]

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 later) and that a number of free and commercial tools are available. Such tools are typically optimized 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.
Footnotes

2There are a few DLs that permit n-aries, which are the $\mathcal{DLR}$ and $\mathcal{CFD}$ families of logics that were inspired by conceptual data modelling and relational models.

3It is the same as the model-theoretic semantics we have seen, but then restated for DLs.

4i.e., the unique name assumption (UNA) means that different names refer to different individuals, as is customary in the database world. There are consequences for UNA/no-UNA both regarding computational complexity and automated reasoners: no-UNA is more costly and the deductions are different when reasoning with UNA or not. We will see the effects of especially the latter in the exercises of Chapter 4.

5It does so ontologically, to which we shall return in Block II.

6Although there are a few interesting things that could be expressed with $\langle U \rangle$, such as concept products [RKH08a], tool support is rarely sufficient for using this feature in practice.

7A 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.