

Foundations of Description Logics

Sebastian Rudolph

Institute AIFB, Karlsruhe Institute of Technology, DE
rudolph@kit.edu

Abstract. This chapter accompanies the foundational lecture on Description Logics (DLs) at the 7th Reasoning Web Summer School in Galway, Ireland, 2011. It introduces basic notions and facts about this family of logics which has significantly gained in importance over the recent years as these logics constitute the formal basis for today's most expressive ontology languages, the OWL (Web Ontology Language) family.

We start out from some general remarks and examples demonstrating the modeling capabilities of description logics as well as their relation to first-order predicate logic. Then we begin our formal treatment by introducing the syntax of DL knowledge bases which comes in three parts: RBox, TBox and ABox. Thereafter, we provide the corresponding standard model-theoretic semantics and give a glimpse of the alternative way of defining the semantics via an embedding into first-order logic with equality.

We continue with an overview of the naming conventions for DLs before we delve into considerations about different notions of semantic likeness (concept and knowledge base equivalence as well as emulation). These are crucial for investigating the expressivity of DLs and performing normalization. We move on by reviewing knowledge representation capabilities brought about by different DL features and their combinations as well as some model-theoretic properties associated thereto.

Subsequently, we consider typical reasoning tasks occurring in the context of DL knowledge bases. We show how some of these tasks can be reduced to each other, and have a look at different algorithmic approaches to realize automated reasoning in DLs.

Finally, we establish connections between DLs and OWL. We show how DL knowledge bases can be expressed in OWL and, conversely, how OWL modeling features can be translated into DLs.

In our considerations, we focus on the description logic *SR_{OIQ}* which underlies the most recent and most expressive yet decidable version of OWL called OWL 2 DL. We concentrate on the logical aspects and omit data types as well as extralogical features from our treatise. Examples and exercises are provided throughout the chapter.

1 Introduction

*Come join the DL vaudeville show!
It's variable-free, although
With quantifiers, not, and, or
Quite deeply rooted in FOLklore.
Still, curing the first-order ailment
We sport decidable entailment!*

While formal, logic-based approaches to representing and working with knowledge occur throughout human history, the advent and widespread adoption of programmable computing devices in the 20th century has led to intensified studies of both theoretical and practical aspects of knowledge representation and automated reasoning. Rooted in early AI approaches, Description Logics (DLs) have developed into one of the main knowledge representation formalisms. The maturity of the field is also reflected by the adoption of description logics as prior specification paradigm for ontological descriptions – culminating in the standardization of the OWL web ontology language by the World Wide Web Consortium (W3C) – as well as the availability of highly optimized and readily deployable (yet open source) tools for automated inferencing. Thanks to this “dissemination path,” DLs constitute the theoretical backbone for information systems in many disciplines, among which life sciences can be seen as the “early adopters” [Sidhu *et al.*, 2005; Wolstencroft *et al.*, 2005; Golbreich *et al.*, 2006].

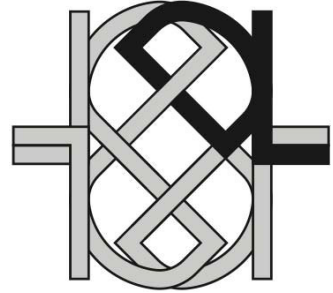


Fig. 1. The DL logo

1.1 Outlook

What is in this Lecture. This document is supposed to give a gentle introduction into state-of-the-art description logics. Before going into technicalities the remainder of this section will briefly discuss how DLs are positioned in the landscape of knowledge representation formalisms, provide some examples for modeling features of DLs, and sketch the most prominent application context: the Semantic Web.

Section 2 starts the formal treatment by introducing the syntax of knowledge bases of the description logic *SROIQ*. Section 3 provides the corresponding model-theoretic semantics and substantiates the claimed connection between DLs and first-order predicate logic (FOL) by giving a translation from *SROIQ* into FOL with equality.

Section 4 reviews the naming scheme for DLs between the basic DL *ALC* and the high-end DL *SROIQ*. Section 5 provides several notions that capture that different syntactic specifications may have the same (or “alike”) semantical impact. The motivation of Section 6 is to give a feeling for the modeling power provided by different constructs and the according model-theoretic consequences.

Subsequently, Section 7 considers typical reasoning tasks normally occurring in the context of DL-based knowledge representation and discusses the mutual

reducibility of these tasks. In Section 8, we give a shallow overview over different algorithmic paradigms for automated inferencing with DLs. Finally, in Section 9, we provide a way to translate *SRITQ* knowledge bases into OWL ontologies and, conversely, show how OWL axioms can be translated into DLs.

What is not in this Lecture. Due to space limitations, we have to restrict this lecture in many respects. We will focus on the core logical aspects of description logics and hence omit datatypes, keys, etc. despite their obvious practical importance for knowledge representation. Likewise, this is not supposed to be an introduction into OWL nor any other Semantic Web specification language. Thus, we will only briefly state how DL knowledge bases can be translated into OWL such that OWL reasoning tools can be harnessed to perform DL reasoning tasks. Moreover, we will refrain from looking into sub-Boolean fragments of DLs, even though they are practically important for serving as theoretical basis for the tractable profiles of the latest version of OWL. On the theoretical side, we will omit considerations about computational complexity of reasoning tasks.

Required Previous Knowledge. This lecture is meant to be introductory and foundational. Consequently, we tried to make it as self-contained as feasibly possible and hope that it is comprehensible even without any background in formal logics, although it can do no harm either. We presume, however, a certain familiarity with basic concepts and notations of naïve set theory. We do not expect prior knowledge about Semantic Web formalisms like the Resource Description Framework (RDF) or OWL, still it would come handy to fully comprehend the comments about the connections between DLs and OWL.

1.2 DLs in the Context of Other Formalisms

Historically, DLs have emerged from semantic networks [Quillian, 1968] and frame-based systems [Minsky, 1974]. These early knowledge representation approaches had the advantage of being rather intuitively readable and comprehensible. On the downside, it turned out that the understanding of the precise meaning of these diagrammatic representations differed widely amongst humans. This also became apparent by the heterogeneous behavior of tools implemented to reason with these structures. Under a plethora of names (among them *terminological systems* and *concept languages*), description logics developed out of the attempt to endow these intuitive representations with a formal semantics to establish a common ground for human and tool interoperability.

With the formal semantics introduced it was rather immediately clear that – abstracting from the syntax used – DLs can be seen as a fragment of first-order predicate logic (short: FOL), many of them even as a fragment of FOL’s two-variable fragment [Borgida, 1996] in cases extended with counting quantifiers [Pratt-Hartmann, 2005]. As opposed to general FOL where logical inferencing is undecidable, DL research has been focusing on decidable fragments to such an extent that today, decidability is almost conceived as a necessary condition to call a formalism a DL.

Remark 1. Recap that in theoretical computer science, a class of problems is called decidable, if there is a generic algorithm that can take any problem instance from this class as an input and provide a yes-or-no answer to it after finite time. In the context of logics, the generic problem normally investigated is whether a given set of statements logically entails another statement. In case there is no danger of confusion about the type of problem considered, sometimes the logic itself is called decidable or undecidable.

In contrast to the well-known correspondence to FOL, it took some time to discover the close relation of DLs to modal logics [Schild, 1991]; in fact, the basic description logic \mathcal{ALC} is just a syntactic variant of the multi-modal logic \mathbf{K}_m . As a consequence of this, there is also a close relationship of DLs to the Guarded Fragment [Andréka *et al.*, 1998], a very expressive fragment of FOL which is still decidable.

For application purposes, DLs can be tailored to the specific requirements of a concrete usage scenario. To this end, a set of modeling features is selected such that the resulting logic has sufficient expressivity for the intended purpose while still being manageable in terms of the inferencing needed. This strategy has led to thorough investigations and finally a deeper understanding of the impact of the diverse standard modeling features on decidability and complexity of reasoning.

Remark 2. Thereby, the boundaries of the above mentioned fragments are sometimes crossed. For instance, functionality statements and cardinality constraints in general are not supported by the Guarded Fragment, the same holds for transitivity statements, which also lie outside the two-variable fragment. DLs featuring regular expressions on roles [Calvanese *et al.*, 2009] even go beyond FOL with equality, but we will not discuss them here.

Beyond decidability, a crucial design principle in DLs is to establish favorable trade-offs between expressivity and scalability. On the theoretical side, establishing complexity results for inferencing problems (a tradition started by Brachman and Levesque [1984] and meanwhile widely accepted as central part of the DL research methodology) helps to roughly estimate how scalable and how “implementable” reasoning methods are likely to be. Of course, for the deployment in practice, many engineering and optimization considerations are necessary even if they do not influence the worst-case complexities. Today, there exist several highly optimized and efficient systems for reasoning in DL-based formalisms [Motik *et al.*, 2009c; Sirin *et al.*, 2007; Tsarkov and Horrocks, 2006].

1.3 DL Modeling in a Nutshell

This section provides an informal introduction of the most common modeling features in DLs. For the interested reader with some background in logics, we will relate them to FOL with equality by giving the corresponding terms and logical translations in square brackets.

All DLs are based on a vocabulary [signature] containing individual names [constants], concept names [unary predicates] and role names [binary predicates].

Two specific class names, \top and \perp , denote the concept containing all individuals and the empty concept, respectively. Usually, a DL knowledge base [theory] is partitioned into an assertional part, called ABox and a terminological part, which is further subdivided into TBox and RBox. The ABox contains assertional knowledge [ground facts], the notation of which coincides with FOL: there are concept assertions such as

$$\mathbf{Actor}(\mathbf{angelina})$$

(indicating that the individual named **angelina** belongs to the set of all actors) and role assertions like

$$\mathbf{married}(\mathbf{angelina}, \mathbf{brad})$$

(stating that the individuals named **angelina** and **brad** are in the relation of being married). The TBox contains universal statements. The notation used in DLs does not need variables and is inspired by set theory. We can specify subsumptions, e.g. by expressing that every actor is an artist via

$$\mathbf{Actor} \sqsubseteq \mathbf{Artist}$$

$[\forall x(\mathbf{Actor}(x) \rightarrow \mathbf{Artist}(x))]$. A specific feature of DLs is that concept names can be combined into complex concepts by Boolean operators, as in

$$\mathbf{Actor} \sqcap \mathbf{USGovernor} \sqsubseteq \mathbf{Bodybuilder} \sqcup \neg \mathbf{Austrian}$$

$[\forall x(\mathbf{Actor}(x) \wedge \mathbf{USGovernor}(x) \rightarrow \mathbf{Bodybuilder}(x) \vee \neg \mathbf{Austrian}(x))]$, expressing that every actor who is a US governor is also a bodybuilder or not Austrian. Another way to define complex concepts is by quantifying over roles, as for instance in

$$\exists \mathbf{knows}.\mathbf{Actor} \sqsubseteq \forall \mathbf{hasfriend}.\mathbf{Envious}$$

$[\forall x(\exists y(\mathbf{knows}(x, y) \wedge \mathbf{Actor}(y)) \rightarrow \forall z(\mathbf{hasfriend}(x, z) \rightarrow \mathbf{Envious}(z)))]$, which states that everybody knowing some actor has only envious friends.

The modeling features introduced above constitute \mathcal{ALC} (*attributive language with complements*, [Schmidt-Schauß and Smolka, 1991]), the smallest DL that is Boolean-closed (i.e. it allows Boolean operators to be applied to concepts without restriction).

As stated before, in order to satisfy requirements emerging from practical modeling scenarios, these basic modeling features have been enriched by more and more expressive features for specifying and querying knowledge. In DLs, this development has led from the basic \mathcal{ALC} to more expressive formalisms. *Role inverses* can be used to “traverse” roles backward e.g. in

$$\exists \mathbf{HasChild}.\top \sqsubseteq \forall \mathbf{hasChild}^{\neg}.\mathbf{Grandparent}$$

$[\forall x(\exists y(\mathbf{hasChild}(x, y)) \rightarrow \forall z(\mathbf{hasChild}(z, x) \rightarrow \mathbf{Grandparent}(x)))]$, expressing that everybody having a child is the child of only grandparents. *Cardinality constraints* allow for specifying the number of related instances:

$$\mathbf{Polygamist} \sqsubseteq \geq 2.\mathbf{Married}.\top$$

$[\forall x(\text{Polygamist}(x) \rightarrow \exists y \exists z(\text{Married}(x, y) \wedge \text{Married}(x, z) \wedge y \neq z))]$ states that a polygamist is married to at least two distinct individuals. By means of *nominals*, classes can be defined by enumerating their instances: the axiom

$$\exists \text{Married}.\{\text{brad}\} \sqsubseteq \{\text{angelina}\}$$

$[\exists x(\text{Married}(x, \text{brad}) \rightarrow x = \text{angelina})]$ claims that being married to Brad is a property only applying to Angelina.

The RBox of a DL knowledge base allows for further, role-centric modeling features. These include role inclusion statements as for instance:

$$\text{married} \sqsubseteq \text{loves}$$

$[\forall x \forall y(\text{married}(x, y) \rightarrow \text{loves}(x, y))]$, which states that being married to somebody implies loving them. A more general and expressive variant of role inclusions are role-chain axioms as in

$$\text{hasChild}^- \circ \text{hasChild} \sqsubseteq \text{hasSibling}$$

$[\forall x \forall y \forall z(\text{hasChild}(y, x) \wedge \text{hasChild}(y, z) \rightarrow \text{hasSibling}(x, z))]$, saying that the child of somebody I am a child of is my sibling.

1.4 The Semantic Web

The rise of the World Wide Web as a large body of digitally accessible knowledge has inspired a plethora of research related to the question how to organize and formalize knowledge on the Web in order to allow for automated, intelligent retrieval and combination of the stored information. The term *Semantic Web* stands for a variety of research and standardization efforts towards this goal, and DLs constitute a crucial part of this endeavor. The underlying idea of the Semantic Web is to provide information on the Web in a sufficiently formal and structured way to enable “intelligent” processing by machines. To this end, several key requirements can be identified: First, it is necessary to agree on common and open standards for representing information, in order to enable the exchange of information between diverse applications and platforms and subsequently the combination of pieces of information from different origins. Such standards have to be defined in a clear formal way but at the same time, they need to be flexible and extendable.

In fact, the World Wide Web Consortium (W3C) has fostered and approved the definition of the basic Semantic Web standards. The ontology languages RDF and its extension RDF Schema (RDFS) as well as OWL have been deliberately developed for a deployment in the Semantic Web.¹

¹ Originating from philosophy, the term *ontology* is not precisely defined in the computer science context either and a lot of deviating definitions can be found throughout the literature. In this treatise, we will use the term to simply refer to a document created in RDF(S) or OWL, modeling knowledge of an application domain. Thereby, we will consider it to be equivalent with the arguably more appropriate term *knowledge base*.

As the second key ingredient for the Semantic Web, methods are needed which automatically infer new knowledge from given knowledge. In order to maximally benefit from specified knowledge, it must be possible to obtain information that is not explicitly given but constitutes a logical consequence of what is known. This directly leads to the multifarious field of formal logic, and in particular to the area of automated reasoning. A significant portion of DL research has been spawned by problems and usage scenarios from the Semantic Web area.

2 Syntax

Deluxe DL delivery

Will come in boxes (number: three),

Precisely marked with A, T, R.

The first exhibits solid grounding,

The next allows for simple counting,

The third one's strictly regular.

In this section, we provide the definition of the expressive description logic *SR_{OIQ}* [Horrocks *et al.*, 2006] which serves as the logical basis for OWL 2 DL, the most expressive member of the OWL family where inferencing is still decidable. Most of today's mainstream DLs are, in fact, sublanguages of *SR_{OIQ}*.

DLs are based on three disjoint sets of primal elements:

- The set N_I of *individual names* contains all names used to denote singular entities (be they persons, objects or anything else) in our domain of interest. Examples would be `brad`, `excalibur`, `rhine`, or `sun`.
- The set N_C of *concept names* contains names that refer to types, categories, or classes of entities, usually characterized by common properties. Typical concept names are `Mammal`, `Country`, `Organization`, but also `Yellow` or `English`.
- The set N_R of *role names* contains names that denote binary relationships which may hold between individuals of a domain, for instance: `marriedWith`, `fatherOf`, `likes`, or `locatedIn`.

Remark 3. There are no mandatory rules for writing and typography of vocabulary elements. According to a convention most widely adopted, we capitalize concept names whereas individual and role names are written in lower case. Moreover, camel case is used for names corresponding to multi word units in natural language.

Having these name sets at hand (they are usually jointly referred to as *vocabulary* or *signature*), we can now turn to the three building blocks of *SR_{OIQ}* knowledge bases: RBox, TBox and ABox.

2.1 RBox

A *SR_{OIQ}* RBox captures interdependencies between the roles of the considered knowledge base. Given the set N_R of role names, a *role* is either the *universal*

role u or it has the form \mathbf{r} or \mathbf{r}^- for any role name \mathbf{r} . The set of roles will be denoted by \mathbf{R} . For convenience, we introduce the function Inv that “inverts” roles, i.e. we set $Inv(\mathbf{r}) := \mathbf{r}^-$ and $Inv(\mathbf{r}^-) := \mathbf{r}$ in order to simplify notation. In the sequel, we will use the symbols r, s , possibly with subscripts, to denote roles.

A *role inclusion axiom* (RIA, sometimes also referred to as *role chain axiom*) is a statement of the form $r_1 \circ \dots \circ r_n \sqsubseteq r$ where r_1, \dots, r_n, r are roles. As a special case thereof (for $n = 1$), we obtain *simple role inclusions* $r \sqsubseteq s$. Typical examples of role inclusion axioms are $\mathbf{owns} \circ \mathbf{partOf} \sqsubseteq \mathbf{owns}$ or $\mathbf{fatherOf} \sqsubseteq \mathbf{childOf}^-$. A finite set of such RIAs is called a *role hierarchy*.

Given a role hierarchy, it is useful to distinguish the roles that can be “created” by role chains of length greater than one from those which cannot. Consequently, we define *non-simple roles* as follows:

- S1. Every role r occurring in a RIA $r_1 \circ \dots \circ r_n \sqsubseteq r$ where $n > 1$ is non-simple.
- S2. Every role r occurring in a simple role inclusion $s \sqsubseteq r$ with a non-simple s is itself non-simple.
- S3. If r is non-simple then so is $Inv(r)$.
- S4. No other role is non-simple.

We let \mathbf{R}^n denote the set of all non-simple roles of a role hierarchy and call all the other roles *simple* denoted by $\mathbf{R}^s = \mathbf{R} \setminus \mathbf{R}^n$.

Example 4. Consider the following role hierarchy:

$$\mathbf{motherOf} \sqsubseteq \mathbf{parentOf} \tag{1}$$

$$\mathbf{parentOf} \sqsubseteq \mathbf{ancestorOf} \tag{2}$$

$$\mathbf{ancestorOf} \circ \mathbf{ancestorOf} \sqsubseteq \mathbf{ancestorOf} \tag{3}$$

$$\mathbf{ancestorOf} \sqsubseteq \mathbf{descendantOf}^- \tag{4}$$

Then we can use S1. to find that $\mathbf{ancestorOf}$ is non-simple due to (3). This allows us to conclude that $\mathbf{descendantOf}^-$ is non-simple via (4) and S2.. From the above follows via S3. that also $\mathbf{ancestorOf}^-$ and $\mathbf{descendantOf}$ must be non-simple. Finally, S4. ensures that $\mathbf{motherOf}$, $\mathbf{motherOf}^-$, $\mathbf{parentOf}$, and $\mathbf{parentOf}^-$ are simple.

In order to ensure decidability of the ensuing logic, we cannot allow arbitrary role hierarchies but have to restrict to those which have the property of being *regular*. Formally, a role hierarchy is regular if there is a strict partial order \prec on the non-simple roles \mathbf{R}^n such that

- $S \prec R$ iff $Inv(S) \prec R$, and
- every RIA is of one of the forms

$$\mathbf{R1} \quad r \circ r \sqsubseteq r,$$

$$\mathbf{R2} \quad Inv(r) \sqsubseteq r,$$

$$\mathbf{R3} \quad s_1 \circ \dots \circ s_n \sqsubseteq r,$$

$$\mathbf{R4} \quad r \circ s_1 \circ \dots \circ s_n \sqsubseteq r,$$

$$\mathbf{R5} \quad s_1 \circ \dots \circ s_n \circ r \sqsubseteq r,$$

such that $r \in \mathbf{N}_R$ is a (non-inverse) role name \mathbf{r} , and $s_i \prec r$ for $i = 1, \dots, n$ whenever s_i is non-simple.

Example 5. Consider the following role hierarchy containing the RIAs: $s \circ s \sqsubseteq s$, $r \circ s \sqsubseteq r$, and $r \circ s \circ r \sqsubseteq t$. First observe that all involved atomic roles are non-simple. If we define \prec such that $s^- \prec s \prec r^- \prec r \prec t^- \prec t$, then all the above criteria are satisfied: the first RIA is an instance of R1, the second is an instance of R4, and the third is an instance of R3. Hence this role hierarchy is regular.

Example 6. Assume a role hierarchy containing $r \circ t \circ s \sqsubseteq t$ as the only axiom. Only t is non-simple here, still this role hierarchy is not regular, as the RIA does not fit any of the allowed forms R1–R5 (to see this, note that \prec is required to be strict, therefore $t \not\prec t$ must always be the case, irrespective of the concrete choice of \prec).

Example 7. Let a role hierarchy contain the two RIAs $r \circ s \sqsubseteq s$, and $s \circ r \sqsubseteq r$. While each of these RIAs alone would be acceptable as a role hierarchy, they do not go well together: the first requires $r \prec s$ (due to R5) whereas the second enforces $s \prec r$ (due to R4) which as a whole violates the condition of \prec being a strict order. Thus the considered role hierarchy is not regular.

A *role characteristic* is a statement of the form $\text{Ref}(r)$ (*reflexivity*), $\text{Asy}(s)$ (*asymmetry*), or $\text{Dis}(s, s')$ (*role disjointness*), where s and s' are simple roles while r may be simple or non-simple. A *SROIQ RBox* (usually denoted by \mathcal{R}) is the union of a finite set of role characteristics together with a role hierarchy. A *SROIQ RBox* is regular if its role hierarchy is regular.

2.2 TBox

Given a *SROIQ RBox* \mathcal{R} as defined in the previous section, we now inductively define *concept expressions* (also simply called *concepts*) as follows:

- every concept name $C \in \mathbf{N}_C$ is a concept expression,
- \top and \perp are concept expressions, called *top concept* and *bottom concept*, respectively,
- $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ is a concept expression for every finite set $\{\mathbf{a}_1, \dots, \mathbf{a}_n\} \subseteq \mathbf{N}_I$ of individual names; concepts of this type are called *nominal concepts*,
- if C and D are concept expressions then so are $\neg C$ (*negation*), $C \sqcap D$ (*intersection*), $C \sqcup D$ (*union*),
- if r is a role and C is a concept expression, then $\exists r.C$ (*existential quantification*) and $\forall r.C$ (*universal quantification*) are also concept expressions,
- if r is a simple role, n is a natural number and C is a concept expression, then $\exists r.\text{Self}$ (*self restriction*), $\geq nr.C$ (*at-least restriction*), and $\leq nr.C$ (*at-most restriction*) are also concept expressions. The latter two are also jointly referred to as *qualified number restrictions* or *cardinality constraints*.

We will denote the set of all concept expressions thus defined by \mathbf{C} . Throughout this chapter, the symbols C, D will be used to denote concept expressions.

Remark 8. Note that the definition of concept expressions depends on the underlying RBox due to the restriction of some concept expressions to contain only simple roles.

A *general concept inclusion axiom* (short: GCI) has the form $C \sqsubseteq D$ where C and D are concepts. This kind of statement is also sometimes called *subsumption axiom*, as $C \sqsubseteq D$ is often read “ C is subsumed by D .” Sometimes, this axiom type is also referred to as *is-a relationship*, inspired by the often chosen wording for this type of statement (e.g. “a cat is a mammal” would be a typical verbalization of $\text{Cat} \sqsubseteq \text{Mammal}$).

Remark 9. Sometimes, $C \sqsubseteq D$ is also called a *subconcept* statement with $C \sqsubseteq D$ being read “ C is a subconcept of D .” While this is well justified by standard formal theories of (human) conceptual thinking where concepts are hierarchically ordered by subconcept-superconcept relationships [Ganter and Wille, 1997], this naming is unfortunate in the DL setting since it can also be understood syntactically to mean subformula of a concept term. Thus we do not use this term and whenever referring to the latter meaning, we speak of *subexpressions* of a concept.

Finally, a *SRQIQ TBox* (usually denoted by \mathcal{T}) is a finite set of GCIs.

2.3 ABox

The *ABox* of a knowledge base contains information that applies to single individuals as opposed to the GCIs in the TBox, which represent statements which are generally true for all individuals alike.

An *individual assertion* can have any of the following forms:

- $C(\mathbf{a})$, called *concept assertion*,
- $r(\mathbf{a}, \mathbf{b})$, called *role assertion*,
- $\neg r(\mathbf{a}, \mathbf{b})$, called *negated role assertion*,
- $\mathbf{a} \approx \mathbf{b}$, called *equality statement*, or
- $\mathbf{a} \not\approx \mathbf{b}$, called *inequality statement*,

with $\mathbf{a}, \mathbf{b} \in \mathbf{N}_I$ individual names, $C \in \mathbf{C}$ a concept expression, and $r \in \mathbf{R}$ a role.

Remark 10. Of course, also the form $\neg C(\mathbf{a})$ is captured by the above definition since $\neg C$ is again a concept expression, as opposed to roles, which do not allow for negation (note that the inverse of a role is something quite different from its negation).

A *SRQIQ ABox* (usually denoted by \mathcal{A}) is a finite set of individual assertions. We call an ABox *extensionally reduced* if the only concepts and roles occurring therein are concept names and roles names, respectively.

Remark 11. It should be noted that the separation between ABox and TBox – originally conceived for less expressive DLs – becomes less sharp once nominal concepts are allowed, since nominal concepts allow for referring to single individuals in the TBox as well. In fact, every of the different types of individual assertions can be expressed by a GCI featuring nominals: $C(\mathbf{a})$ becomes $\{\mathbf{a}\} \sqsubseteq C$, $(\neg)r(\mathbf{a}, \mathbf{b})$ is equivalent to $\{\mathbf{a}\} \sqsubseteq (\neg)\exists r.\{\mathbf{b}\}$, $\mathbf{a} \approx \mathbf{b}$ can be rewritten into $\{\mathbf{a}\} \sqsubseteq \{\mathbf{b}\}$, and $\mathbf{a} \not\approx \mathbf{b}$ into $\{\mathbf{a}\} \sqsubseteq \neg\{\mathbf{b}\}$. Still the distinction is not entirely meaningless even for DLs featuring nominals as soon as *data complexity* of reasoning is investigated.

A *SR_{OIQ}* knowledge base \mathcal{KB} is the union of a regular RBox \mathcal{R} and a TBox \mathcal{T} as well as an ABox \mathcal{A} for \mathcal{R} . The elements of \mathcal{KB} are referred to as *axioms*. Given a knowledge base \mathcal{KB} we write $N_I(\mathcal{KB})$, $N_C(\mathcal{KB})$, and $N_R(\mathcal{KB})$ to denote those individual names, concept names, and role names which occur in \mathcal{KB} , respectively.

Example 12. As an example consider the following knowledge base \mathcal{KB} :

RBox \mathcal{R}
owns \sqsubseteq caresFor “If somebody owns something, they care for it.”
TBox \mathcal{T}
Healthy \sqsubseteq \neg Dead “Healthy beings are not dead.”
Cat \sqsubseteq Dead \sqcup Alive “Every cat is dead or alive.”
HappyCatOwner \sqsubseteq \exists owns.Cat \sqcap \forall caresFor.Healthy “A happy cat owner owns a cat and all beings he cares for are healthy.”
ABox \mathcal{A}
HappyCatOwner (schrödinger) “Schrödinger is a happy cat owner.”

3 Semantics

*Semantics has wide applications
To relationship-based altercations,
For semantics unveils
What a statement entails
Depending on interpretations.*

Like for any other logic, the definition of a formal semantics for DLs boils down to providing a consequence relation that determines whether an axiom logically follows from (also: is entailed by) a given set of axioms. The semantics of description logics is defined in a model-theoretic way. Thereby, one central notion is that of an interpretation. Interpretations might be conceived as potential “realities” or “worlds.” In particular, interpretations need in no way comply with the actual reality.

3.1 Interpretations

In the case of DLs, an interpretation, normally denoted with \mathcal{I} , provides

- a nonempty set $\Delta^{\mathcal{I}}$, called the *domain* or also *universe of discourse* which can be understood as the entirety of individuals or things existing in the “world” that \mathcal{I} represents, and

- a function $\cdot^{\mathcal{I}}$, called *interpretation function* which connects the vocabulary elements (i.e., the individual, concept, and role names) to $\Delta^{\mathcal{I}}$, by providing
 - for each individual name $a \in N_I$ a corresponding individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$ from the domain,
 - for each concept name $A \in N_C$ a corresponding set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ of domain elements (as opposed to the domain itself, $A^{\mathcal{I}}$ is allowed to be empty), and
 - for each role name $r \in N_R$ a corresponding (also possibly empty) set $r^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ of ordered pairs of domain elements.

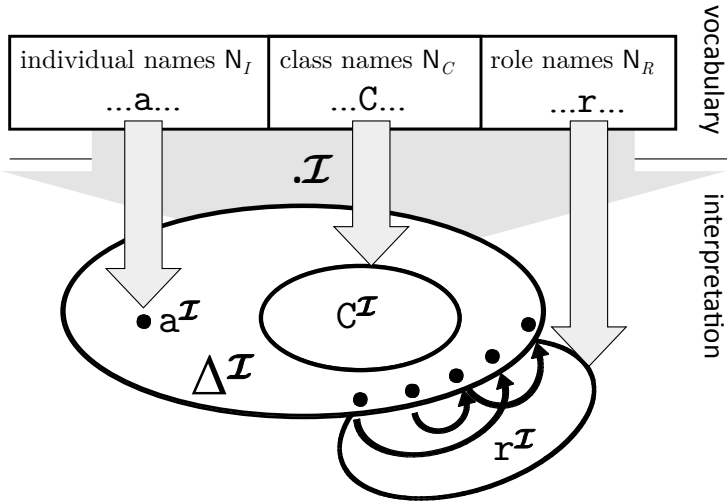


Fig. 2. Structure of DL interpretations

Figure 2 depicts this definition graphically. For domain elements $\delta, \delta' \in \Delta$, the intuitive meaning of $\delta \in A^{\mathcal{I}}$ is that the individual δ belongs to the class described by the concept name A , while $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$ means that δ is connected to δ' by the relation denoted by the role name r .

Remark 13. To avoid confusion, it is important to strictly separate syntactic notions (referring to the vocabulary and axioms) from the semantic notions (referring to the domain and domain elements). Individual names, concept names and role names are syntactic entities and so are roles and concepts. Individuals are elements of $\Delta^{\mathcal{I}}$ and hence semantic entities. In order to refer to the semantic counterparts of concepts and roles, one would use the terms *concept extension* or *role extension*, respectively. Single elements of the extension of a concept or role are also called *concept instances* or *role instances*.

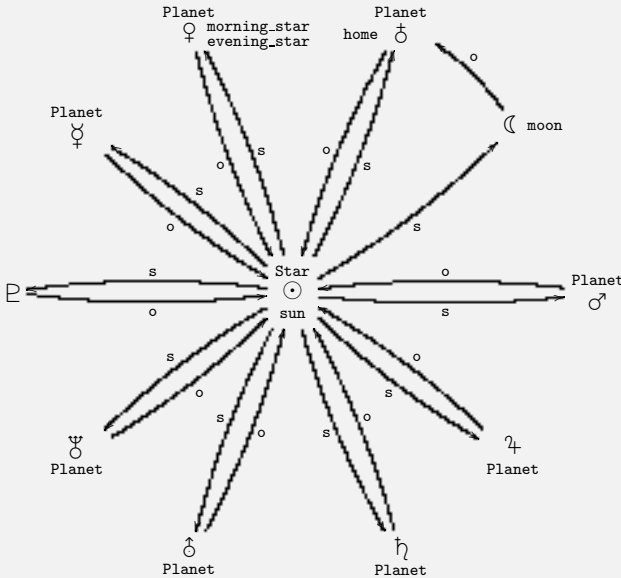
Example 14. Consider the following signature:

- $N_I = \{\text{sun, morning_star, evening_star, moon, home}\}$.
- $N_C = \{\text{Planet, Star}\}$.
- $N_R = \{\text{orbitsAround, shinesOn}\}$.

We now define an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as follows: Let our domain $\Delta^{\mathcal{I}}$ contain the following elements: $\odot, \wp, \varphi, \delta, \mathbb{C}, \sigma, \mathcal{A}, \mathfrak{h}, \delta, \mathfrak{P}, \mathbb{E}$. We define the interpretation function by

$$\begin{array}{ll}
 \text{sun}^{\mathcal{I}} = \odot & \text{Planet}^{\mathcal{I}} = \{\wp, \varphi, \delta, \sigma, \mathcal{A}, \mathfrak{h}, \delta, \mathfrak{P}\} \\
 \text{morning_star}^{\mathcal{I}} = \varphi & \text{Star}^{\mathcal{I}} = \{\odot\} \\
 \text{evening_star}^{\mathcal{I}} = \varphi & \text{orbitsAround}^{\mathcal{I}} = \{\langle \wp, \odot \rangle, \langle \varphi, \odot \rangle, \langle \delta, \odot \rangle, \langle \sigma, \odot \rangle, \langle \mathcal{A}, \odot \rangle, \\
 \text{moon}^{\mathcal{I}} = \mathbb{C} & \langle \mathfrak{h}, \odot \rangle, \langle \delta, \odot \rangle, \langle \mathfrak{P}, \odot \rangle, \langle \mathbb{E}, \odot \rangle, \langle \mathbb{C}, \delta \rangle\} \\
 \text{home}^{\mathcal{I}} = \delta & \text{shinesOn}^{\mathcal{I}} = \{\langle \odot, \wp \rangle, \langle \odot, \varphi \rangle, \langle \odot, \delta \rangle, \langle \odot, \mathbb{C} \rangle, \langle \odot, \sigma \rangle, \\
 & \langle \odot, \mathcal{A} \rangle, \langle \odot, \mathfrak{h} \rangle, \langle \odot, \delta \rangle, \langle \odot, \mathfrak{P} \rangle, \langle \odot, \mathbb{E} \rangle\}
 \end{array}$$

For a better understanding, it is often helpful to display an interpretation as a directed graph with labeled nodes and arcs. Thereby, the nodes correspond to the domain individuals $\Delta^{\mathcal{I}}$ where a node $\delta \in \Delta^{\mathcal{I}}$ gets labeled by the individual names assigned to it (i.e. those $\mathbf{a} \in N_I$ for which $\mathbf{a}^{\mathcal{I}} = \delta$) as well as the concept names \mathbf{A} in the extensions of which δ lies (i.e. $\delta \in \mathbf{A}^{\mathcal{I}}$). Moreover, whenever a pair of two domain individuals $\delta, \delta' \in \Delta^{\mathcal{I}}$ is in the extension of a role name \mathbf{r} (that is, if $\langle \delta, \delta' \rangle \in \mathbf{r}^{\mathcal{I}}$), a directed arc is drawn from δ to δ' and labeled with \mathbf{r} . The graphical representation of the interpretation \mathcal{I} defined above would then look like this (where we abbreviate `orbitsAround` by `o` and `shinesOn` by `s`):



Remark 15. One should keep in mind that the domain $\Delta^{\mathcal{I}}$ is not required to be finite, but can also be an infinite set. It is also possible to consider only interpretations with finite domains, but then one explicitly talks about *finite models* or *finite satisfiability*. There are logics where infinite interpretations are “dispensable” as there are always finite ones that do the same job, these logics are said to have the *finite model property*. *SROIQ* does not have this property. However, since DLs are normally fragments of first-order logic, we can safely restrict our attention to interpretations with countable domains (that is, domains having at most as many individuals as there are natural numbers). This is a consequence of the downward part of the Theorem of Löwenheim-Skolem, according to which every FOL theory that has an arbitrary infinite model also has a countable one.

Example 16. As an example of an interpretation, this time with an infinite domain, consider the following vocabulary:

- $N_I = \{\text{zero}\}$.
- $N_C = \{\text{Prime}, \text{Positive}\}$.
- $N_R = \{\text{hasSuccessor}, \text{lessThan}, \text{multipleOf}\}$.

Now, we define \mathcal{I} as follows: let $\Delta^{\mathcal{I}} = \mathbb{N} = \{0, 1, 2, \dots\}$, i.e., the set of all natural numbers including zero. Furthermore, we let $\text{zero}^{\mathcal{I}} = 0$, as well as $\text{Prime}^{\mathcal{I}} = \{n \mid n \text{ is a prime number}\}$ and $\text{Positive}^{\mathcal{I}} = \{n \mid n > 0\}$. For the roles, we define

- $\text{hasSuccessor}^{\mathcal{I}} = \{\langle n, n + 1 \rangle \mid n \in \mathbb{N}\}$
- $\text{lessThan}^{\mathcal{I}} = \{\langle n, n' \rangle \mid n < n', n, n' \in \mathbb{N}\}$
- $\text{multipleOf}^{\mathcal{I}} = \{\langle n, n' \rangle \mid \exists k. n = k \cdot n', n, n', k \in \mathbb{N}\}$

Note that this interpretation is well defined, although it has an infinite domain. For space reasons, we refrain from providing the corresponding graph representation.

Remark 17. Note that the definition of an interpretation does not require that different individual names denote different individuals, that is, it may happen that for two individual names \mathbf{a} and \mathbf{b} , we have $\mathbf{a}^{\mathcal{I}} = \mathbf{b}^{\mathcal{I}}$. A stronger definition of DL interpretations that excludes such cases is usually referred to as *unique name assumption* (short: UNA). Note also, that not every domain element $\delta \in \Delta$ needs to be named, i.e., there may be δ for which no individual name \mathbf{a} with $\mathbf{a}^{\mathcal{I}} = \delta$ exists. For obvious reasons, such individuals are usually referred to as *anonymous individuals*.

3.2 Satisfaction of Axioms

By now, we have seen that an interpretation determines the semantic counterparts of vocabulary elements. However, in order to finally determine the truth of complex axioms, it is necessary to also find the counterparts of complex concepts and roles. We provide a definition according to which the semantics of a complex language construct can be obtained from the semantics of its constituents (thereby following the principle of *compositional semantics*). Formally, this is done by “lifting” the interpretation function $\cdot^{\mathcal{I}}$ to these complex expressions.

First we extend the interpretation function from role names to roles by letting $u^{\mathcal{I}} = \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ (that is: the universal role interconnects any two individuals of the domain and also every individual with itself), and assigning to inverted role names r^- the set of all pairs $\langle \delta, \delta' \rangle$ of domain elements for which $\langle \delta', \delta \rangle$ is contained in $r^{\mathcal{I}}$.

Next we define the interpretation function for concepts:

- \top is the concept which is true for every individual of the domain, hence $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$.
- \perp is the concept which has no instances, hence $\perp^{\mathcal{I}} = \emptyset$.
- $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ is the concept containing exactly the individuals denoted by $\mathbf{a}_1, \dots, \mathbf{a}_n$, therefore $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}^{\mathcal{I}} = \{\mathbf{a}_1^{\mathcal{I}}, \dots, \mathbf{a}_n^{\mathcal{I}}\}$
- $\neg C$ is supposed to denote the set of all those domain individuals that are not contained in the extension of C , i.e., $(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$.
- $C \sqcap D$ is the concept comprising all individuals that are simultaneously in C and D , thus we define $(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$.
- $C \sqcup D$ contains individuals being present in C or D (or both), therefore we let $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$.
- $\exists r.C$ is the concept that holds for an individual $\delta \in \Delta^{\mathcal{I}}$ exactly if there is some domain individual $\delta' \in \Delta^{\mathcal{I}}$ such that δ is connected to δ' via the relation denoted by r and δ' belongs to the extension of the concept C , formally: $(\exists r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \exists \delta' \in \Delta^{\mathcal{I}}. (\langle \delta, \delta' \rangle \in r^{\mathcal{I}} \wedge \delta' \in C^{\mathcal{I}})\}$.
- $\forall r.C$ denotes the set of individuals $\delta \in \Delta^{\mathcal{I}}$ with the following property: whenever δ is connected to some domain individual $\delta' \in \Delta^{\mathcal{I}}$ via the relation denoted by r , then δ' belongs to the extension of the concept C , formally: $(\forall r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \forall \delta' \in \Delta^{\mathcal{I}}. (\langle \delta, \delta' \rangle \in r^{\mathcal{I}} \rightarrow \delta' \in C^{\mathcal{I}})\}$.
- $\exists r.\text{Self}$ comprises those domain individuals which are r -related to themselves, thus we let $(\exists r.\text{Self})^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \langle x, x \rangle \in r^{\mathcal{I}}\}$.
- $\leq n r.C$ refers to the domain elements $\delta \in \Delta^{\mathcal{I}}$ for which no more than n individuals exist to which δ is r -related and that are in the extension of C , formally: $(\leq n r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \#\{\delta' \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \wedge \delta' \in C^{\mathcal{I}}\} \leq n\}$ (thereby $\#S$ is used to denote the cardinality of a set S , i.e., the number of its elements).
- $\geq n r.C$, dual to the case before, denotes those domain elements having at least n such r -related elements: $(\geq n r.C)^{\mathcal{I}} = \{\delta \in \Delta^{\mathcal{I}} \mid \#\{\delta' \in \Delta^{\mathcal{I}} \mid \langle \delta, \delta' \rangle \in r^{\mathcal{I}} \wedge \delta' \in C^{\mathcal{I}}\} \geq n\}$.

Remark 18. The reader should be aware that by the above definition, the extension of the concept $\forall r.C$ contains every domain individual $\delta \in \Delta^{\mathcal{I}}$ that is not r -connected to any δ' . For instance, the concept `forallChild.Happy` comprises all individuals all of whose children are happy (alternatively, and arguably less confusing: all individuals that do not have children which are not happy). This includes those individuals not having children at all. In fact, when modeling with DLs, the concept $\forall r.\perp$ is often used to refer to all individuals not being r -connected to any other individual (nor to themselves).

Example 19. Consider the interpretation \mathcal{I} from Example 14. With the lifting of the interpretation function just defined, we are able to determine the extension of concepts and roles as follows:

$$\begin{aligned}
& \text{orbitsaround}^{-\mathcal{I}} \\
&= \{\langle \odot, \wp \rangle, \langle \odot, \varphi \rangle, \langle \odot, \delta \rangle, \langle \odot, \sigma \rangle, \langle \odot, \tau \rangle, \langle \odot, \eta \rangle, \langle \odot, \delta \rangle, \langle \odot, \wp \rangle, \langle \odot, \rho \rangle\} \\
& (\forall \text{orbitsAround.}(\neg \text{Star}))^{\mathcal{I}} \\
&= \{\delta \mid \forall \delta'. (\langle \delta, \delta' \rangle \in \text{orbitsAround}^{\mathcal{I}} \rightarrow \delta' \in (\neg \text{Star})^{\mathcal{I}})\} \\
&= \{\delta \mid \forall \delta'. (\langle \delta, \delta' \rangle \in \text{orbitsAround}^{\mathcal{I}} \rightarrow \delta' \in \Delta^{\mathcal{I}} \setminus \text{Star}^{\mathcal{I}})\} \\
&= \{\delta \mid \forall \delta'. (\langle \delta, \delta' \rangle \in \text{orbitsAround}^{\mathcal{I}} \rightarrow \delta' \in \{\wp, \varphi, \delta, \sigma, \tau, \eta, \delta, \wp, \rho\})\} \\
&= \{\odot, \mathbb{C}\} \\
& ((\neg \text{Planet}) \sqcap \exists \text{orbitsAround. Star})^{\mathcal{I}} \\
&= (\neg \text{Planet})^{\mathcal{I}} \cap (\exists \text{orbitsAround. Star})^{\mathcal{I}} \\
&= \Delta^{\mathcal{I}} \setminus \text{Planet}^{\mathcal{I}} \cap \{\delta \mid \exists \delta'. (\langle \delta, \delta' \rangle \in \text{orbitsAround}^{\mathcal{I}} \wedge \delta' \in \text{Star}^{\mathcal{I}})\} \\
&= \{\odot, \mathbb{C}, \rho\} \cap \{\wp, \varphi, \delta, \sigma, \tau, \eta, \delta, \wp, \rho\} \\
&= \{\rho\} \\
& (\geq 2 \text{shinesOn.}\{\text{morning_star}, \text{evening_star}\})^{\mathcal{I}} \\
&= \{\delta \mid \#\{\delta' \mid \langle \delta, \delta' \rangle \in \text{shinesOn}^{\mathcal{I}} \wedge \delta' \in \{\text{morning_star}, \text{evening_star}\}^{\mathcal{I}}\} \geq 2\} \\
&= \{\delta \mid \#\{\delta' \mid \langle \delta, \delta' \rangle \in \text{shinesOn}^{\mathcal{I}} \wedge \delta' \in \{\text{morning_star}^{\mathcal{I}}, \text{evening_star}^{\mathcal{I}}\}\} \geq 2\} \\
&= \{\delta \mid \#\{\delta' \mid \langle \delta, \delta' \rangle \in \text{shinesOn}^{\mathcal{I}} \wedge \delta' \in \{\varphi\}\} \geq 2\} \\
&= \emptyset
\end{aligned}$$

Exercise 1. Describe – both verbally and formally – the extension of the following concepts with respect to the interpretation \mathcal{I} defined in Example 16:

- $\forall \text{hasSuccessor}^{-}.\text{Positive}$
- $\exists \text{multipleOf}.\text{Self}$
- $\exists \text{multipleOf}.\exists \text{hasSuccessor}^{-}.\exists \text{hasSuccessor}^{-}.\{\text{zero}\}$
- $\geq 10 \text{lessThan}^{-}.\text{Prime}$
- $\neg \text{Prime} \sqcap \leq 2 \text{multipleOf}.\top$
- $\exists \text{lessThan}.\text{Prime}$
- $\forall \text{multipleOf}.\{\text{zero}\} \sqcup \exists \text{multipleOf}.\exists \text{hasSuccessor}^{-}.\exists \text{hasSuccessor}^{-}.\{\text{zero}\}$

The final purpose of the (lifted) interpretation function is to determine satisfaction of axioms. In the following, we define when an axiom α is true (also: holds), given a specific interpretation \mathcal{I} . If this is the case, we also say that \mathcal{I} is a *model* of α or that \mathcal{I} *satisfies* α and write $\mathcal{I} \models \alpha$.

- A role inclusion axiom $r_1 \circ \dots \circ r_n \sqsubseteq r$ holds in \mathcal{I} if for every sequence $\delta_0, \dots, \delta_n \in \Delta^{\mathcal{I}}$ for which holds $\langle \delta_0, \delta_1 \rangle \in r_1^{\mathcal{I}}, \dots, \langle \delta_{n-1}, \delta_n \rangle \in r_n^{\mathcal{I}}$, also $\langle \delta_0, \delta_n \rangle \in r^{\mathcal{I}}$ is satisfied. Figuratively, this means that every path in $\Delta^{\mathcal{I}}$ that traverses the roles r_1, \dots, r_n (in the given order) must have a direct

r -“shortcut.” When using \circ as symbol for the relation product, we can write down this condition as $r_1^{\mathcal{I}} \circ \dots \circ r_n^{\mathcal{I}} \subseteq r^{\mathcal{I}}$.

- A role disjointness statement $\text{Dis}(r, s)$ is true in \mathcal{I} if every two domain individuals $\delta, \delta' \in \Delta^{\mathcal{I}}$ that are connected via an r -relation are not connected via an s -relation. In other words, we can say that the two roles are mutually exclusive which can be formally expressed by the condition $r^{\mathcal{I}} \cap s^{\mathcal{I}} = \emptyset$.
- A general concept inclusion $C \sqsubseteq D$ is satisfied by \mathcal{I} , if every instance of C is also an instance of D . An alternative wording of this would be that the extension of C is contained in the extension of D , formally $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$.
- A concept assertion $C(\mathbf{a})$ holds in \mathcal{I} if the individual with the name \mathbf{a} is an instance of the concept C , that is $\mathbf{a}^{\mathcal{I}} \in C^{\mathcal{I}}$.
- A role assertion $r(\mathbf{a}, \mathbf{b})$ is true in \mathcal{I} if the individual denoted by \mathbf{a} is r connected to the individual denoted by \mathbf{b} , i.e. the extension of r contains the corresponding pair of domain elements: $\langle \mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$.
- \mathcal{I} is a model of $\neg r(\mathbf{a}, \mathbf{b})$ exactly if it is not a model of $r(\mathbf{a}, \mathbf{b})$.
- The equality statement $\mathbf{a} \approx \mathbf{b}$ holds in \mathcal{I} if the individual names \mathbf{a} and \mathbf{b} refer to the same domain individual, i.e. $\mathbf{a}^{\mathcal{I}} = \mathbf{b}^{\mathcal{I}}$.
- \mathcal{I} is a model of $\mathbf{a} \not\approx \mathbf{b}$ exactly if it is not a model of $\mathbf{a} \approx \mathbf{b}$.

Example 20. We now check for some example axioms whether interpretation \mathcal{I} from Example 14 satisfies them.

- `morning_star` \approx `evening_star` is true, since `morning_star` ^{\mathcal{I}} = \varnothing = `evening_star` ^{\mathcal{I}} , i.e. the two names denote the same domain individual.
- `orbitsAround` \circ `orbitsAround` \sqsubseteq `shinesOn`⁻ is also true: The only chain of domain individuals $\delta_1, \delta_2, \delta_3$ with $\langle \delta_1, \delta_2 \rangle \in \text{orbitsAround}^{\mathcal{I}}$ and $\langle \delta_2, \delta_3 \rangle \in \text{orbitsAround}^{\mathcal{I}}$ is $\delta_1 = \mathbb{C}$, $\delta_2 = \delta$, $\delta_3 = \odot$. Therefore, we obtain `orbitsAround` ^{\mathcal{I}} \circ `orbitsAround` ^{\mathcal{I}} = $\{ \langle \mathbb{C}, \odot \rangle \}$. On the other hand, due to $\langle \odot, \mathbb{C} \rangle \in \text{shinesOn}^{\mathcal{I}}$ we obtain $\langle \mathbb{C}, \odot \rangle \in \text{shinesOn}^{-\mathcal{I}}$.
- `Star`(`evening_star`) is false since the domain element `evening_star` ^{\mathcal{I}} = \varnothing is not contained in `Star` ^{\mathcal{I}} = $\{ \odot \}$.
- `Planet` \sqsubseteq $\neg \{ \text{sun}, \text{moon} \}$ is valid in \mathcal{I} as we get $(\neg \{ \text{sun}, \text{moon} \})^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus (\{ \text{sun}, \text{moon} \})^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus \{ \odot, \mathbb{C} \} = \{ \varnothing, \varphi, \delta, \sigma, \mathfrak{A}, \mathfrak{H}, \delta, \mathfrak{P}, \mathbb{E} \}$ which is a superset of `Planet` ^{\mathcal{I}} = $\{ \varnothing, \varphi, \delta, \sigma, \mathfrak{A}, \mathfrak{H}, \delta, \mathfrak{P} \}$.
- `shinesOn`(`moon`, `earth`) does not hold in \mathcal{I} since the pair of the respective individuals is not contained in the extension of the `shinesOn` role: `moon` ^{\mathcal{I}} , `earth` ^{\mathcal{I}} = $\langle \mathbb{C}, \delta \rangle \notin \text{shinesOn}^{\mathcal{I}}$.
- $\top \sqsubseteq \forall \text{shinesOn}^{-} . \{ \text{sun} \}$ is true. To see this we first need to find $(\forall \text{shinesOn}^{-} . \{ \text{sun} \})^{\mathcal{I}}$. In words, this concept comprises those objects that are shone upon by nothing but the sun (if they are shone upon by anything at all). Formally, to check whether a domain individual δ is in the extension of that concept, we have to verify that every individual δ' with $\langle \delta, \delta' \rangle \in \text{shinesOn}^{-\mathcal{I}}$ (which is equivalent to $\langle \delta', \delta \rangle \in \text{shinesOn}^{\mathcal{I}}$) also satisfies $\delta' \in \{ \text{sun} \}^{\mathcal{I}}$ which just means $\delta' = \odot$. Scrutinizing all elements of $\Delta^{\mathcal{I}}$, we find this condition satisfied for each, therefore we have $\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} = (\forall \text{shinesOn}^{-} . \{ \text{sun} \})^{\mathcal{I}}$.
- $\text{Dis}(\text{orbitsAround}, \text{shinesOn})$ is satisfied by \mathcal{I} since no pair $\langle \delta, \delta' \rangle$ is contained in the extensions of both `orbitsAround` and `shinesOn` and therefore `orbitsAround` ^{\mathcal{I}} \cap `shinesOn` ^{\mathcal{I}} = \emptyset .

Exercise 2. Decide whether the following axioms are satisfied by the interpretation \mathcal{I} from Example 16.

- (a) $\text{hasSuccessor} \sqsubseteq \text{lessThan}$
- (b) $\exists \text{hasSuccessor}^- . \exists \text{hasSuccessor}^- . \{\text{zero}\} \sqsubseteq \text{Prime}$
- (c) $\top \sqsubseteq \forall \text{multipleOf}^- . \{\text{zero}\}$
- (d) $\text{Dis}(\text{divisibleBy}, \text{lessThan}^-)$
- (e) $\text{multipleOf} \circ \text{multipleOf} \sqsubseteq \text{multipleOf}$
- (f) $\top \sqsubseteq \leq \text{lhasSuccessor.Positive}$
- (g) $\text{zero} \not\approx \text{zero}$
- (h) $\leq \text{lmultipleOf}^- . \top(\text{zero})$
- (i) $\top \sqsubseteq \forall \text{lessThan} . \exists \text{lessThan} . (\text{Prime} \sqcap \exists \text{hasSuccessor} . \exists \text{hasSuccessor} . \text{Prime})$

Now that we have defined when an interpretation \mathcal{I} is a model of an axiom, we can easily extend this notion to whole knowledge bases: we say that \mathcal{I} is a *model* of a given knowledge base \mathcal{KB} (also: \mathcal{I} *satisfies* \mathcal{KB} , written $\mathcal{I} \models \mathcal{KB}$), if it satisfies all the axioms of \mathcal{KB} , i.e., if $\mathcal{I} \models \alpha$ for every $\alpha \in \mathcal{KB}$. Moreover, a knowledge base \mathcal{KB} is called *satisfiable* or *consistent* if it has a model, and it is called *unsatisfiable* or *inconsistent* or *contradictory* otherwise.

Example 21. The following knowledge base is inconsistent.

$\text{Reindeer} \sqcap \exists \text{hasNose.Red}(\text{rudolph})$	$\text{Reindeer} \sqsubseteq \text{Mammal}$
$\forall \text{worksFor}^- . (\neg \text{Reindeer} \sqcup \text{Flies})(\text{santa})$	$\text{Mammal} \sqcap \text{Flies} \sqsubseteq \text{Bat}$
$\text{worksFor}(\text{rudolph}, \text{santa})$	$\text{Bat} \sqsubseteq \forall \text{worksFor} . \{\text{batman}\}$
$\text{santa} \not\approx \text{batman}$	

Remark 22. Note that, for determining whether a knowledge base satisfies an interpretation \mathcal{I} , only the value of $\cdot^{\mathcal{I}}$ for those individual, concept, and role names are relevant, that occur in \mathcal{KB} . All vocabulary elements not contained in $\mathbf{N}_I(\mathcal{KB}) \cup \mathbf{N}_C(\mathcal{KB}) \cup \mathbf{N}_R(\mathcal{KB})$ can be mapped arbitrarily and do not influence the semantics.

3.3 Logical Consequence

So far, we have defined a “modelhood” relation, which for a given interpretation and a given set of axioms determines whether the axiom is true with respect to the interpretation. Remember that the actual purpose of a formal semantics is to provide a consequence relation, which tells us whether an axiom is a logical consequence of a knowledge base. This consequence relation is commonly also denoted by \models and defined as follows: an axiom α is a *consequence* of (also *entailed by*) a knowledge base \mathcal{KB} (written: $\mathcal{KB} \models \alpha$) if every model of \mathcal{KB} is also a model of α , i.e. for every \mathcal{I} with $\mathcal{I} \models \mathcal{KB}$ also holds $\mathcal{I} \models \alpha$.

Remark 23. As a straightforward consequence of this model-theoretic definition of consequences we obtain the fact that an inconsistent knowledge base entails any axiom, since the considered set of models which have to satisfy the axiom is empty and hence the condition is vacuously true. This effect, well-known in many logics, is called the *principle of explosion* according to which “anything follows from a contradiction.”

Exercise 3. *Decide whether the following propositions about the knowledge base \mathcal{KB} from Example 12 are true and give evidence:*

- (a) \mathcal{KB} is satisfiable,
- (b) $\mathcal{KB} \models \text{Alive}(\text{schrödinger})$,
- (c) $\mathcal{KB} \models \text{Dead} \sqcap \text{Alive} \sqsubseteq \perp$,
- (d) $\mathcal{KB} \models \text{Alive} \sqsubseteq \text{Healthy}$.

Exercise 4. *Decide whether the following statements are true or false and justify your decision. For arbitrary \mathcal{SROIQ} knowledge bases \mathcal{KB} and \mathcal{KB}' holds:*

- (a) *If an axiom α is a logical consequence of the empty knowledge base, i.e. $\emptyset \models \alpha$, then it is the consequence of any other knowledge base \mathcal{KB} .*
- (b) *The larger a knowledge base, the more models it has. That is, if $\mathcal{KB} \subseteq \mathcal{KB}'$ then every model of \mathcal{KB} is also a model of \mathcal{KB}' .*
- (c) *The larger a knowledge base, the more consequences it has. That is, if $\mathcal{KB} \subseteq \mathcal{KB}'$ then every logical consequence from \mathcal{KB} is a logical consequence from \mathcal{KB}' .*
- (d) *If $\neg C(\mathbf{a}) \in \mathcal{KB}$, then $\mathcal{KB} \models C(\mathbf{a})$ can never hold (for arbitrary concepts C).*
- (e) *If two knowledge bases are different ($\mathcal{KB} \neq \mathcal{KB}'$), then they also differ in terms of logical consequences, i.e., there is an axiom α such that $\mathcal{KB} \models \alpha$ and $\mathcal{KB}' \not\models \alpha$ or vice versa.*

3.4 Excursus: Semantics via Embedding into FOL

As mentioned before, it is often said that most description logics, including \mathcal{SROIQ} , are fragments of first-order predicate logic (FOL). Technically, this statement is somewhat misleading since, from a syntax point of view, most DL axioms are not FOL formulae. What is rather meant by this statement is the following: It is obvious that DL interpretations have the same structure as FOL interpretations if one conceives individual names as constants, concept names as unary predicates and role names as binary predicates. Under this assumption, one can define an easy syntactic translation τ which, applied to a DL axiom α , yields a FOL sentence $\tau(\alpha)$ such that the model sets of α and $\tau(\alpha)$ coincide, that is an interpretation \mathcal{I} is a model of α exactly if it is a model of $\tau(\alpha)$. Consequently, every reasoning problem in a DL is easily transferrable to an equivalent reasoning problem in FOL, whence the semantics of description logics could – as an alternative to the previously introduced way – be defined by reducing it to the semantics of FOL via the mentioned translation.

Remark 24. Obviously, the converse cannot be the case, for any decidable DL: supposing it were, we could decide any FOL reasoning problem by translating it to the DL and then deciding the DL version. This clearly contradicts the well-known undecidability of FOL.

We provide here a definition of τ but omit a proof of its correctness. More precisely, the translation outputs first-order predicate logic with equality, a mild generalization of pure first-order predicate logic featuring an equality predicate

=. Every *SRONTQ* knowledge base \mathcal{KB} thus translates via τ to a theory $\tau(\mathcal{KB})$ in first-order predicate logic with equality. We define

$$\tau(\mathcal{KB}) = \bigcup_{\alpha \in \mathcal{KB}} \tau(\alpha),$$

i.e., we translate every axiom of the knowledge base separately into a FOL sentence. How exactly $\tau(\alpha)$ is defined depends on the type of the axiom α .

However, first we have to define auxiliary translation functions $\tau_{\mathbf{R}} : \mathbf{R} \times \mathbf{Var} \times \mathbf{Var} \rightarrow \text{FOL}$ for roles and $\tau_{\mathbf{C}} : \mathbf{C} \times \mathbf{Var} \rightarrow \text{FOL}$ for concepts (where $\mathbf{Var} = \{x_0, x_1, \dots\}$ is a set of variables):

$$\begin{aligned} \tau_{\mathbf{R}}(u, x_i, x_j) &= \mathbf{true} \\ \tau_{\mathbf{R}}(\mathbf{r}, x_i, x_j) &= \mathbf{r}(x_i, x_j) \\ \tau_{\mathbf{R}}(\mathbf{r}^-, x_i, x_j) &= \mathbf{r}(x_j, x_i) \\ \\ \tau_{\mathbf{C}}(\mathbf{A}, x_i) &= \mathbf{A}(x_i) \\ \tau_{\mathbf{C}}(\top, x_i) &= \mathbf{true} \\ \tau_{\mathbf{C}}(\perp, x_i) &= \mathbf{false} \\ \tau_{\mathbf{C}}(\{\mathbf{a}_1, \dots, \mathbf{a}_n\}, x_i) &= \bigvee_{1 \leq j \leq n} x_i = \mathbf{a}_j \\ \tau_{\mathbf{C}}(\neg C, x_i) &= \neg \tau_{\mathbf{C}}(C, x_i) \\ \tau_{\mathbf{C}}(C \sqcap D, x_i) &= \tau_{\mathbf{C}}(C, x_i) \wedge \tau_{\mathbf{C}}(D, x_i) \\ \tau_{\mathbf{C}}(C \sqcup D, x_i) &= \tau_{\mathbf{C}}(C, x_i) \vee \tau_{\mathbf{C}}(D, x_i) \\ \tau_{\mathbf{C}}(\exists r.C, x_i) &= \exists x_{i+1}. (\tau_{\mathbf{R}}(r, x_i, x_{i+1}) \wedge \tau_{\mathbf{C}}(C, x_{i+1})) \\ \tau_{\mathbf{C}}(\forall r.C, x_i) &= \forall x_{i+1}. (\tau_{\mathbf{R}}(r, x_i, x_{i+1}) \rightarrow \tau_{\mathbf{C}}(C, x_{i+1})) \\ \tau_{\mathbf{C}}(\exists r.\text{Self}, x_i) &= \tau_{\mathbf{R}}(r, x_i, x_i) \\ \tau_{\mathbf{C}}(\geq nr.C, x_i) &= \exists x_{i+1} \dots x_{i+n}. \left(\bigwedge_{i+1 \leq j < k \leq i+n} (x_j \neq x_k) \right. \\ &\quad \left. \wedge \bigwedge_{i+1 \leq j \leq i+n} (\tau_{\mathbf{R}}(r, x_i, x_j) \wedge \tau_{\mathbf{C}}(C, x_j)) \right) \\ \tau_{\mathbf{C}}(\leq nr.C, x_i) &= \neg \tau_{\mathbf{C}}(\geq (n+1)r.C, x_i) \end{aligned}$$

Obviously, the translation assigns to a role a FOL formula with (at most) two free variables and to a concept a FOL formula with (at most) one free variable. Now we are ready to translate axioms:

$$\begin{aligned} \tau(r_1 \circ \dots \circ r_n \sqsubseteq r) &= \forall x_0 \dots x_n. \left(\bigwedge_{1 \leq i \leq n} \tau_{\mathbf{R}}(r_i, x_{i-1}, x_i) \right) \rightarrow \tau_{\mathbf{R}}(r, x_0, x_n) \\ \tau(\text{Dis}(r, r')) &= \forall x_0 x_1. (\tau_{\mathbf{R}}(r, x_0, x_1) \rightarrow \neg \tau_{\mathbf{R}}(r', x_0, x_1)) \\ \tau(C \sqsubseteq D) &= \forall x_0. (\tau_{\mathbf{C}}(C, x_0) \rightarrow \tau_{\mathbf{C}}(D, x_0)) \\ \tau(C(\mathbf{a})) &= \tau_{\mathbf{C}}(C, x_0)[x_0/\mathbf{a}] \\ \tau(r(\mathbf{a}, \mathbf{b})) &= \tau_{\mathbf{R}}(r, x_0, x_1)[x_0/\mathbf{a}][x_1/\mathbf{b}] \\ \tau(\neg r(\mathbf{a}, \mathbf{b})) &= \neg \tau(r(\mathbf{a}, \mathbf{b})) \\ \tau(a \approx b) &= a = b \\ \tau(a \not\approx b) &= \neg(a = b) \end{aligned}$$

Exercise 5. Translate the axioms from Example 20 and Exercise 2 into first-order logic with equality.

Remark 25. The considerations in this section do not apply to all DLs, since also extensions of DLs with non-first-order features have been defined and investigated such as non-monotonic features, regular expressions as role constructors or fixpoint operators. However, the mainstream DLs for which mature reasoners exist and which have been used as a basis for OWL are all first-order-embeddable.

4 Description Logics Nomenclature

*What's in a name? That which we call, say, SHIQ,
By any other name would do the trick.
While DL names might leave the novice SHOQed,
Some principles of ALCHemy unlocked
Enable understanding in a minute:
Though it be madness, yet there's method in it.*

There is a well-established naming convention for DLs. The naming scheme for mainstream DLs can be summarized as follows:

$$((\mathcal{ALC} | \mathcal{S})[\mathcal{H}] | \mathcal{SR})[\mathcal{O}][\mathcal{I}][\mathcal{F} | \mathcal{N} | \mathcal{Q}]$$

The meaning of the name constituents is as follows:

- \mathcal{ALC} is an abbreviation for *attributive language with complements* [Schmidt-Schauß and Smolka, 1991]. This DL disallows RBox axioms as well as the universal role, role inverses, cardinality constraints, nominal concepts, and self concepts.
- By \mathcal{S} we denote \mathcal{ALC} where we additionally allow transitivity statements, i.e., specific role chain axioms of the shape $\mathbf{r} \circ \mathbf{r} \sqsubseteq \mathbf{r}$ for $\mathbf{r} \in \mathbb{N}_R$. The name goes back to the name of a modal logic called \mathcal{S} .
- \mathcal{ALC} and \mathcal{S} can be extended by role hierarchies (obtaining \mathcal{ALCH} or \mathcal{SH}) which allow for simple role inclusions, i.e., role chain axioms of the shape $r \sqsubseteq s$.
- \mathcal{SR} denotes \mathcal{ALC} extended with all kinds of RBox axioms as well as self concepts.
- The letter \mathcal{O} in the name of a DL indicates that nominal concepts are supported.
- When a DL contains \mathcal{I} then it features role inverses.
- The letter \mathcal{F} at the end of a DL name enables support for role functionality statements which can be expressed as $\top \sqsubseteq \leq 1.\top$.
- \mathcal{N} at the end of a DL name allows for unqualified number restrictions, i.e., concepts of the shape $\geq nr.\top$ and $\leq nr.\top$.
- \mathcal{Q} indicates support for arbitrary qualified number restrictions.

As becomes clear from the previous descriptions, \mathcal{S} contains \mathcal{ALC} . Moreover \mathcal{SR} subsumes all of \mathcal{ALC} , \mathcal{ALCH} , \mathcal{S} , and \mathcal{SH} . Finally \mathcal{F} becomes obsolete once \mathcal{N} is present and both are superseded by \mathcal{Q} .

Exercise 6. *Come up with a partial order diagram displaying syntactic containment of all DLs that match the above naming scheme and do not contain \mathcal{F} or \mathcal{N} .*

Exercise 7. *Name, for each of the following knowledge bases, the “smallest” DL that contains it:*

- (a) *the knowledge base from Example 12,*
- (b) *the knowledge base from Example 21,*
- (c) *the knowledge base consisting of the axioms (a), (b) and (e) from Exercise 2,*
- (d) *the knowledge base containing the axioms*
 $\top \sqsubseteq \exists \text{sameAs}.\text{Self}$ $\top \sqsubseteq \leq 1 \text{sameAs}.\top$ $\text{batman} \sqsubseteq \neg \exists \text{sameAs}^-. \{ \text{santa} \}.$

5 Equivalences, Emulation, Normalization

*Don't give told consequences lip,
 Nor 'bout equivalences quip,
 'Cause often it's the formal norm
 That statements be in normal form.*

The language of the DL *SR₀IQ* is rather redundant, that is, a matter can be formulated in in many ways that are syntactically different but semantically the same. In the following, we will survey different kinds of “semantical alikeness.” Moreover we also discuss how this “syntactic redundancy” can be reduced by reverting to so-called normal forms, which come handy for preprocessing knowledge bases before performing actual automated reasoning, but are also useful to alleviate proof work when certain meta-logical properties have to be shown.

5.1 Concept Equivalences

A very basic form of “semantical alikeness” is *concept equivalence*. Two concepts $C, D \in \mathbf{C}$ are called equivalent – which is usually denoted by $C \equiv D$ – if they have the same extension in any interpretation \mathcal{I} , i.e. $C^{\mathcal{I}} = D^{\mathcal{I}}$. Note that this notion does not presume a fixed knowledge base, thus it really refers to all possible interpretations \mathcal{I} .

Remark 26. It is easy to see that the definition of concept equivalence can be reformulated in terms of axiom entailment: $C \equiv D$ holds exactly if the empty knowledge base entails both $C \sqsubseteq D$ and $D \sqsubseteq C$, i.e. $\emptyset \models C \sqsubseteq D$ and $\emptyset \models D \sqsubseteq C$. In fact, sometimes in the literature, statements of the form $C \equiv D$ are allowed to occur in knowledge bases as TBox axioms.

Exercise 8. *Contemplate whether the condition from Remark 26 can be captured by just one axiom, i.e. whether there is an axiom α such that $\emptyset \models \alpha$ if and only if $C \equiv D$. If this question cannot be answered right now, you may revisit it after having read this section.*

Quite a few basic concept equivalences (which are normally simply taken for granted without further consideration) can be directly traced back to the semantics definition for concepts. To recognize and memorize the equivalences it is quite helpful that the syntactical notation of concept constructors (\sqcup, \sqcap) is inspired by the associated set-theoretical interpretation (\cup, \cap) and is also very related to the corresponding notation in propositional logic (\vee, \wedge). First, we find that both concept intersection and union are commutative, associative and idempotent.

$$\begin{array}{lll}
 C \sqcap D \equiv D \sqcap C & C \sqcup D \equiv D \sqcup C & \text{commutativity} \\
 (C \sqcap D) \sqcap E \equiv C \sqcap (D \sqcap E) & (C \sqcup D) \sqcup E \equiv C \sqcup (D \sqcup E) & \text{associativity} \\
 C \sqcap C \equiv C & C \sqcup C \equiv C & \text{idempotency}
 \end{array}$$

The law of associativity alone already releases us from the duty to put parentheses if the union or intersection of more than two concepts is written down, this allows us to write $C \sqcup D \sqcup D$ or $C \sqcap D \sqcap D$ without causing semantical ambiguity due to the missing precedence information. By virtue of the laws of commutativity, associativity, and idempotency together, we can even conceive unions and intersections of many concepts as sets and write for concept sets $\{C_1, \dots, C_n\} = \mathfrak{C} \subseteq \mathbf{C}$

$$\bigsqcup_{C \in \mathfrak{C}} C \quad \text{or} \quad \bigsqcap_{C \in \mathfrak{C}} C$$

instead of $C_1 \sqcup \dots \sqcup C_n$ or $C_1 \sqcap \dots \sqcap C_n$, respectively.

While the aforementioned laws deal with semantical properties of \sqcap and \sqcup separately, the following cope with their mutual interactions. On the right hand side, we see that the two connectives are distributive over each other, while the equivalences on the right are usually referred to as absorption laws.

$$\begin{array}{ll}
 (C \sqcup D) \sqcap E \equiv (C \sqcap E) \sqcup (D \sqcap E) & (C \sqcup D) \sqcap C \equiv C \\
 (C \sqcap D) \sqcup E \equiv (C \sqcup E) \sqcap (D \sqcup E) & (C \sqcap D) \sqcup C \equiv C
 \end{array}$$

Next, we investigate equivalence correspondences involving negation and are certainly not too surprised to find that double negation can be removed and also that the laws of de Morgan are valid in the DL setting:

$$\begin{array}{l}
 \neg\neg C \equiv C \\
 \neg(C \sqcap D) \equiv \neg D \sqcup \neg C \\
 \neg(C \sqcup D) \equiv \neg D \sqcap \neg C
 \end{array}$$

Beyond but similar to the de Morgan laws, negation can be shifted past quantifiers or be “absorbed” by number restrictions and we obtain:

$$\begin{array}{l}
 \neg\exists r.C \equiv \forall r.\neg C \\
 \neg\forall r.C \equiv \exists r.\neg C \\
 \neg\leq nr.C \equiv \geq(n+1)r.C \\
 \neg\geq(n+1)r.C \equiv \leq nr.C
 \end{array}$$

The above laws provide a lot of leeway to move negation around. In particular, they ensure that for every concept there exists a concept in *negation normal form*. A concept is said to be in negation normal form (short: NNF), if the only negation symbols in it occur in front of concept names, nominal concepts or self concepts. Given a concept C , we determine the concept $nmf(C)$ which is in negation normal form and satisfies $C \equiv nmf(C)$ by applying the recursive function nmf :

$$\begin{aligned}
 nmf(C) &:= C \text{ if } C \in \{A, \neg A, \{\mathbf{a}_1, \dots, \mathbf{a}_n\}, \neg\{\mathbf{a}_1, \dots, \mathbf{a}_n\}, \exists r.\text{Self}, \neg\exists r.\text{Self}, \top, \perp\} \\
 nmf(\neg\neg C) &:= nmf(C) \\
 nmf(\neg\top) &:= \perp & nmf(\neg\perp) &:= \top \\
 nmf(C \sqcap D) &:= nmf(C) \sqcap nmf(D) & nmf(\neg(C \sqcap D)) &:= nmf(\neg C) \sqcup nmf(\neg D) \\
 nmf(C \sqcup D) &:= nmf(C) \sqcup nmf(D) & nmf(\neg(C \sqcup D)) &:= nmf(\neg C) \sqcap nmf(\neg D) \\
 nmf(\forall r.C) &:= \forall r.nmf(C) & nmf(\neg\forall r.C) &:= \exists r.nmf(\neg C) \\
 nmf(\exists r.C) &:= \exists r.nmf(C) & nmf(\neg\exists r.C) &:= \forall r.nmf(\neg C) \\
 nmf(\leq n r.C) &:= \leq n r.nmf(C) & nmf(\neg\leq n r.C) &:= \geq (n + 1) r.nmf(C) \\
 nmf(\geq n r.C) &:= \geq n r.nmf(C) & nmf(\neg\geq n r.C) &:= \leq (n - 1) r.nmf(C)
 \end{aligned}$$

The following equivalences show that ≥ 0 cardinality constraints are vacuously true and that existential and universal quantification can be seen as a special case of number restrictions.

$$\begin{aligned}
 \geq 0 r.C &\equiv \top \\
 \geq 1 r.C &\equiv \exists r.C \\
 \leq 0 r.C &\equiv \forall r.\neg C
 \end{aligned}$$

Exercise 9. Argue that for every \mathcal{ALCQ} concept C , there exists a concept C' with $C \equiv C'$ containing (next to concept and role names) only the connectives \neg, \sqcup , and $\geq n$. Provide a function that computes C' .

We finish our enumeration of concept equivalences with some correspondences showing, next to some interactions of quantifiers with \top and \perp , that quantifiers may distribute over corresponding connectives, that nominal concepts can be “split” into unions of singleton nominal concepts, and that in self concepts, inverses don’t make a difference.

$$\begin{aligned}
 \exists r.\perp &\equiv \perp \\
 \forall r.\top &\equiv \top \\
 \exists r.(C \sqcup D) &\equiv \exists r.C \sqcup \exists r.D \\
 \forall r.(C \sqcap D) &\equiv \forall r.C \sqcap \forall r.D \\
 \{\mathbf{a}_1, \dots, \mathbf{a}_n\} &\equiv \{\mathbf{a}_1\} \sqcup \dots \sqcup \{\mathbf{a}_n\} \\
 \exists r^-. \text{Self} &\equiv \exists r. \text{Self}
 \end{aligned}$$

Exercise 10. Give formal proofs for all concept equivalences established in this section.

Exercise 11. Show that the following equivalences are not valid:

- (a) $\exists r.(C \sqcap D) \equiv \exists r.C \sqcap \exists r.D,$
- (b) $C \sqcap (D \sqcup E) \equiv (C \sqcap D) \sqcup E,$
- (c) $\exists r.\{a\} \sqcap \exists r.\{b\} \equiv \geq 2.\{a, b\},$
- (d) $\exists r.\top \sqcap \exists s.\top \equiv \exists r.\exists r^-. \exists s.\top.$

5.2 Knowledge Base Equivalences

Another notion of semantical alikeness is *axiom* or *knowledge base equivalence*. Two knowledge bases \mathcal{KB}_1 and \mathcal{KB}_2 are called equivalent (which we will write $\mathcal{KB}_1 \iff \mathcal{KB}_2$), if their model sets coincide, i.e. if an interpretation \mathcal{I} is a model of \mathcal{KB}_1 exactly if it is a model of \mathcal{KB}_2 . As a special case, we obtain axiom equivalence: α_1 and α_2 are equivalent (written $\alpha_1 \iff \alpha_2$) if the two singleton knowledge bases $\{\alpha_1\}$ and $\{\alpha_2\}$ are equivalent.

In the following, we will review some of the most important knowledge base equivalences which are e.g. used to define knowledge base normal forms. The first two equivalences show that unions on the left hand side as well as intersections on the right hand side of a GCI can be “taken apart” into several axioms. These correspondences are also well known in the logic programming field where they are usually referred to as Lloyd-Topor transformations [Lloyd and Topor, 1984].

$$\begin{aligned} \{A \sqcup B \sqsubseteq C\} &\iff \{A \sqsubseteq C, B \sqsubseteq C\} \\ \{A \sqsubseteq B \sqcap C\} &\iff \{A \sqsubseteq B, A \sqsubseteq C\} \end{aligned}$$

An axiom equivalence also often used for normalization purposes is the following:

$$C \sqsubseteq D \iff \top \sqsubseteq \neg C \sqcup D$$

This allows to transform arbitrary GCIs into the statement that a certain concept (in our case $\neg C \sqcup D$) is “universal”, i.e., that its extension is the whole domain. Moreover, this transformation together with a reverse Lloyd-Topor modification allows to transform an entire TBox into one single universal concept statement.

Example 27. Considering the TBox of the knowledge base from Example 12, we can first perform the following transformations:

- $\text{Healthy} \sqsubseteq \neg \text{Dead}$ becomes $\top \sqsubseteq \neg \text{Healthy} \sqcup \neg \text{Dead}$
- $\text{Cat} \sqsubseteq \text{Dead} \sqcup \text{Alive}$ becomes $\top \sqsubseteq \neg \text{Cat} \sqcup \text{Dead} \sqcup \text{Alive}$
- $\text{HappyCatOwner} \sqsubseteq \exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy}$ becomes $\top \sqsubseteq \neg \text{HappyCatOwner} \sqcup (\exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy})$

Finally, due to the coinciding left hand side of the created GCIs, we can put them together to obtain

$$\begin{aligned} \top \sqsubseteq & (\neg \text{Healthy} \sqcup \neg \text{Dead}) \sqcap (\neg \text{Cat} \sqcup \text{Dead} \sqcup \text{Alive}) \\ & \sqcap (\neg \text{HappyCatOwner} \sqcup (\exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy})) \end{aligned}$$

As already mentioned before, ABox statements can be translated into equivalent TBox statements in any DL that allows for nominals, according to the following equivalences:

$$\begin{aligned}
 C(\mathbf{a}) &\iff \{\mathbf{a}\} \sqsubseteq C \\
 r(\mathbf{a}, \mathbf{b}) &\iff \{\mathbf{a}\} \sqsubseteq \exists r. \{\mathbf{b}\} \\
 \neg r(\mathbf{a}, \mathbf{b}) &\iff \{\mathbf{a}\} \sqsubseteq \neg \exists r. \{\mathbf{b}\} \\
 \mathbf{a} \approx \mathbf{b} &\iff \{\mathbf{a}\} \sqsubseteq \{\mathbf{b}\} \\
 \mathbf{a} \not\approx \mathbf{b} &\iff \{\mathbf{a}\} \sqsubseteq \neg \{\mathbf{b}\}
 \end{aligned}$$

Exercise 12. *It might come as a surprise that the GCI $\{\mathbf{a}\} \sqsubseteq \{\mathbf{b}\}$ is sufficient to express $\mathbf{a} \approx \mathbf{b}$. Argue why the converse inclusion $\{\mathbf{b}\} \sqsubseteq \{\mathbf{a}\}$ is redundant given $\{\mathbf{a}\} \sqsubseteq \{\mathbf{b}\}$.*

In turn this allows to transfer any knowledge base consisting only of an ABox and a TBox into a singular universal concept statement.

Exercise 13. *Consider whether there is a way to also translate RBox axioms into GCIs by a similar technique.*

Example 28. The said equivalences can also be applied reversely and thus used to remove axioms containing nominal concepts from TBoxes. This may be worthwhile doing as nominals in TBoxes normally lead to worse runtimes of reasoning algorithms. Give examples of GCIs containing nominals where this removal is not possible.

The following two equivalences may take a moment to verify intuitively. The essential idea here is to transfer the “standpoint” from the source to the target of a role. These correspondences can be used to remove some inverses from a knowledge base.

$$\begin{aligned}
 \exists r^-. C \sqsubseteq D &\iff C \sqsubseteq \forall r. D \\
 C \sqsubseteq \forall r^-. D &\iff \exists r. C \sqsubseteq D
 \end{aligned}$$

Example 29. Give a formal proof for the two preceding axiom equivalences.

Exercise 14. *Consider whether the inverse can be removed in axioms of the shape $C \sqsubseteq \exists r^-. D$.*

Inverses also give rise to an equivalence between role chain axioms. Intuitively, all roles on both sides of the statement have to be inverted and (which is not really a big surprise) additionally the order of the roles in the chain has to be reverted.

$$r_1 \circ \dots \circ r_n \sqsubseteq r \iff \text{Inv}(r_n) \circ \dots \circ \text{Inv}(r_1) \sqsubseteq \text{Inv}(r)$$

Exercise 15. *In the light of this section, revisit Exercise 7 and discuss how the knowledge bases there could be equivalently rewritten to fit an even “smaller” DL.*

5.3 Emulation

In the previous sections, we considered very strong notions of semantic likeness based on the equality of extensions or model sets, respectively. These notions are symmetric (i.e. they hold both ways) and presume that the signatures used are the same. However, there are certain modeling tasks and certain normalization requirements that can be accomplished only by virtue of additional vocabulary (i.e. auxiliary individual, concept and role names; often those signature elements are called *fresh* in order to indicate that they must not have been used in the knowledge base before).

Example 30. As an easy example, consider the *SRQIQ* axiom $\top \sqsubseteq \exists u.C$, which specifies that the concept C is non-empty, i.e. in every model \mathcal{I} , there must be some individual $\delta \in \Delta^{\mathcal{I}}$ for which $\delta \in C^{\mathcal{I}}$ holds. While we cannot express this equivalently in any DL not featuring the universal role, it is rather easy to do so in an emulating way: we introduce a new individual name c which is meant to denote δ and specify that it denotes an instance of C by the ABox statement $C(c)$. Note that this example also represents a simple form of Skolemisation (which is not the case for all examples of emulation).

This kind of semantic similarity that allows for introducing additional vocabulary is referred to as (*semantic*) *emulation*. Formally, a knowledge base \mathcal{KB}' semantically emulates a knowledge base \mathcal{KB} if the two following conditions hold:

- Every model of \mathcal{KB}' is a model of \mathcal{KB} , formally: given an interpretation \mathcal{I} , we have that $\mathcal{I} \models \mathcal{KB}'$ implies $\mathcal{I} \models \mathcal{KB}$.
- For every model \mathcal{I} of \mathcal{KB} there is a model \mathcal{I}' of \mathcal{KB}' that has the same domain as \mathcal{I} , and coincides with \mathcal{I} on the vocabulary used in \mathcal{KB} . In other words $\Xi^{\mathcal{I}'} = \Xi^{\mathcal{I}}$ for every $\Xi \in \mathbf{N}_I(\mathcal{KB}) \cup \mathbf{N}_C(\mathcal{KB}) \cup \mathbf{N}_R(\mathcal{KB})$.

Remark 31. Note that knowledge base equivalence is a special case of emulation. In particular, every knowledge base emulates itself. Moreover, emulation is transitive: if \mathcal{KB}'' emulates \mathcal{KB}' and \mathcal{KB}' emulates \mathcal{KB} , then \mathcal{KB}'' emulates \mathcal{KB} .

Another common wording for expressing that \mathcal{KB}' emulates \mathcal{KB} is saying that \mathcal{KB}' is *conservative over* \mathcal{KB} . The semantic correspondence between two knowledge bases \mathcal{KB}' and \mathcal{KB} where the former emulates the latter is still quite tight: \mathcal{KB}' is satisfiable exactly if \mathcal{KB} is, the two knowledge bases coincide in terms of entailment for every axiom α which does not use any name from the auxiliary vocabulary used in \mathcal{KB}' , i.e. in this case, we have $\mathcal{KB} \models \alpha$ exactly if $\mathcal{KB}' \models \alpha$. In fact, we even obtain that $\mathcal{KB} \cup \mathcal{KB}_1 \models \mathcal{KB}_2$ exactly if $\mathcal{KB}' \cup \mathcal{KB}_1 \models \mathcal{KB}_2$ for any knowledge bases $\mathcal{KB}_1, \mathcal{KB}_2$ that do not contain any of \mathcal{KB}' 's auxiliary vocabulary. Thus, \mathcal{KB}' can do the same job as \mathcal{KB} in many respects while the possible usage of auxiliary signature elements provides quite some freedom in terms of normalization possibilities.

Example 32. Remember that we call an ABox of a knowledge base extensionally reduced if the only concepts and roles occurring therein are concept names and roles names, respectively. While it is easy to convert an ABox into one not containing statements of the form $\mathbf{r}^-(\mathbf{a}, \mathbf{b})$ (as they can be equivalently expressed by $\mathbf{r}(\mathbf{b}, \mathbf{a})$), concept assertions of the form $C(\mathbf{a})$ where C is not a concept name cannot be removed by equivalent transformations in general. However, by making use of an additional, newly introduced concept name A_C , we can rewrite $C(\mathbf{a})$ into the two axioms $A_C(\mathbf{a})$ and $A_C \sqsubseteq C$ which together do the same job as the original axiom. Thereby, the complex concept is shifted from the ABox into the TBox, whence an exhaustive application of this step to all concept assertions results in a knowledge base \mathcal{KB}' which is extensionally reduced and emulates \mathcal{KB} .

Exercise 16. Prove that $\{A_C(\mathbf{a}), A_C \sqsubseteq C\}$ indeed emulates $\{C(\mathbf{a})\}$.

One normalization being of particular importance for many reasoning algorithms is known under the name *structural reduction*. Essentially, structural reduction aims at reducing the complex structure of axioms by means of introducing concept names for substructures and substituting them. This allows us to omit nestings of role restrictions and boolean operators. Technically, the idea works as follows: let $C[D]$ be a complex concept containing D as a subexpression. Then, we can introduce a fresh concept name A_D and force it to extensionally coincide with D by adding the two axioms $A_D \sqsubseteq D$ and $D \sqsubseteq A_D$ to the knowledge base. This enables us to exchange all occurrences of D in $C[D]$ by A_D , obtaining $C[A_D]$.

Example 33. Consider the axiom

$$\exists \text{livesAt.}\{\text{northPole}\} \sqsubseteq \exists \text{worksFor}^-. (\text{Reindeer} \sqcap \exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny})).$$

Performing structural reduction (and using \equiv as a shortcut for mutual \sqsubseteq) we obtain

$$\begin{aligned} A_{\exists \text{livesAt.}\{\text{northPole}\}} &\sqsubseteq A_{\exists \text{worksFor}^-. (\text{Reindeer} \sqcap \exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny}))} \\ A_{\exists \text{livesAt.}\{\text{northPole}\}} &\equiv \exists \text{livesAt.} A_{\{\text{northPole}\}} \\ A_{\{\text{northPole}\}} &\equiv \{\text{northPole}\} \\ A_{\exists \text{worksFor}^-. (\text{Reindeer} \sqcap \exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny}))} &\equiv \exists \text{worksFor}^-. A_{\text{Reindeer} \sqcap \exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny})} \\ A_{\text{Reindeer} \sqcap \exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny})} &\equiv \text{Reindeer} \sqcap A_{\exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny})} \\ A_{\exists \text{hasNose.}(\text{Red} \sqcap \text{Shiny})} &\equiv \exists \text{hasNose.} A_{\text{Red} \sqcap \text{Shiny}} \\ A_{\text{Red} \sqcap \text{Shiny}} &\equiv \text{Red} \sqcap \text{Shiny} \end{aligned}$$

Remark 34. There are other, more elaborate and space-saving ways to perform structural reduction. In fact normally only one of the two axioms $A_D \sqsubseteq D$ or $D \sqsubseteq A_D$ is necessary to achieve emulation. Which one depends on the position of D inside an axiom related to scopes of negation and other junctors. This position information is captured by the notion of *polarity* of a subexpression.

Exercise 17. Using the technique of structural reduction and other semantic likeness correspondences introduced above, argue that any knowledge base \mathcal{KB} can be emulated by a knowledge base \mathcal{KB}' the TBox of which contains only GCIs of the form

$$\prod_{C \in \mathfrak{C}} C \sqsubseteq \bigsqcup_{D \in \mathfrak{D}} D$$

where $\mathfrak{C} \cup \mathfrak{D}$ contains only concepts of the forms $\{\mathbf{a}\}$, \mathbf{A} , $\exists r.\mathbf{Self}$, $\leq nr.\mathbf{A}$, or $\geq nr.\mathbf{A}$ with $\mathbf{a} \in \mathbf{N}_I$, $\mathbf{A} \in \mathbf{N}_C$ and $r \in \mathbf{R}$ (note that no negation is allowed, whatsoever).

Example 35. Given a concept expression of the form $A \sqsubseteq \geq nr.B$, the cardinality constraint can be removed as follows: We introduce fresh role names r_1, \dots, r_n which we specify as subroles of r (by the axioms $r_i \sqsubseteq r$ for all $1 \leq i \leq n$) and as pairwise disjoint (i.e. we add $\text{Dis}(r_i, r_j)$ for all $1 \leq i < j \leq n$). With that background axiomatization, the above statement can be rewritten into $A \sqsubseteq \prod_{1 \leq i < j \leq n} \exists r_i.B$.

Emulation techniques can also be used to show that a number of statements which can be directly expressed in other logics (such as FOL) but not in DL, are nevertheless expressible by using some “makros” involving auxiliary vocabulary. In the following, we give some examples for this.

The universal role. The universal role u connects all individuals of the described domain. In a DL where this feature is not built in, we may want to introduce a new role u' and write down statements which force u' to behave like the universal role (by making sure that u' must be interpreted as $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ in every model \mathcal{I}). Note that this can be easily done in FOL by the statement $\forall x, y(u'(x, y))$. However, if a DL supports transitivity and nominal concepts, we can obtain the same by introducing a new nominal \mathbf{a}_{aux} and specify the axioms $\top \sqsubseteq \exists u'.\{\mathbf{a}_{aux}\}$ and $\top \sqsubseteq \exists u'^{-}.\{\mathbf{a}_{aux}\}$ and $u' \circ u' \sqsubseteq u'$. The only downside to this is that u' is then necessarily non-simple whence it cannot be used in all places where u could.

Concept products. Sometimes, there are situations where one wants to express that any instance of a concept C is connected with any instance of a concept D via a role r . In fact, *concept product* statements of the form $C \times D \sqsubseteq r$ which express exactly that have been introduced into description logics rather early but never found their way into the mainstream.

Example 36. As an example, the fact that alkaline solutions neutralize acid solutions could be expressed by the concept product axiom $\mathbf{AlkalineSolution} \times \mathbf{AcidSolution} \sqsubseteq \mathbf{neutralises}$.

Again, it is rather easy to find that the FOL statement $\forall x, y(C(x) \wedge D(y) \rightarrow r(x, y))$ realizes this (where we for the sake of simplicity assume that C, D are concept names and r is a role name). However, *SRTOIQ* provides enough modeling capabilities to emulate this situation as well via the GCIs $C \sqsubseteq \exists r_{aux}.\mathbf{Self}$ and $D \sqsubseteq \exists r'_{aux}.\mathbf{Self}$ as well as the complex role inclusion $r_{aux} \circ u \circ r'_{aux} \sqsubseteq r$. Concept products and their impact on reasoning complexity have e.g. been considered by Rudolph *et al.* [2008].

Qualified role inclusion. Likewise, the specialization of roles due to concept memberships of the two involved individuals seems to surpass the modeling capabilities of the DLs treated here. The FOL statement $\forall x, y(C(x) \wedge r(x, y) \wedge D(y) \rightarrow s(x, y))$ (expressing that any C -instance and D -instance that are interconnected by r are also interconnected by s) can be emulated by a DL axiomatization in a similar way as discussed above: Introduce the GCIs $C \sqsubseteq \exists r_{aux}.Self$ and $D \sqsubseteq \exists r'_{aux}.Self$ as well as the complex role inclusion $r_{aux} \circ r \circ r'_{aux} \sqsubseteq s$.

Exercise 18. Use this technique to express the proposition “any person of age having signed a contract which is legal is bound to that contract.” Use the concept names `OfAge`, `Contract`, `Legal` and the role names `hasSigned` and `boundTo`.

Qualified role inclusions and concept products constitute special cases of the more general framework of *description logic rules* as described by Krötzsch *et al.* [2008].

Boolean Combination of Axioms. From the point of view of FOL, it seems quite straightforward that any statement can be negated or any two statements can be connected by disjunction and conjunction, obtaining a new statement inside the logic. In other words, FOL is Boolean-closed on the sentence level. In DLs, the situation is quite different: there is no direct way to, for instance, say that one of the two GCIs $A \sqsubseteq B$ and $C \sqsubseteq D$ must hold. This is, roughly speaking, due to the fact that DL axioms can be understood as “element-wise” propositions (the verbalization of which starts “for each element of the domain holds...”), whereas the above statement gives an alternative choice concerning all individuals at once. Fortunately, *SHOIQ* provides a way to handle this by virtue of the universal role. We first recap that the above axioms can be rewritten into $\top \sqsubseteq \neg A \sqcup B$ and $\top \sqsubseteq \neg C \sqcup D$ respectively. Then we axiomatize the following statement: “every domain element is an instance of $A \sqcup B$ or every domain element is an instance of $C \sqcup D$.” To this end we exploit the fact that every individual is connected to every individual via the universal role, whence we can formally express the above wording by the axiom $\top \sqsubseteq \forall u.(\neg A \sqcup B) \sqcup \forall u.(\neg C \sqcup D)$.

Exercise 19. In fact, the encoding introduced above doesn't need any auxiliary vocabulary. However, arbitrary Boolean combinations of axioms can also be emulated in *SHOIQ*. In that case, the vocabulary must be extended. Explain how this can be done. Hint: try using a “hub nominal.”

Exercise 20. Find a way to emulate $C(\mathbf{a}) \vee D(\mathbf{b})$ in *SHIQ*.

Exercise 21. Consider whether it is possible to emulate *ABox* statements of the shape $\neg r(\mathbf{a}, \mathbf{b})$, $\mathbf{a} \approx \mathbf{b}$, and $\mathbf{a} \not\approx \mathbf{b}$ with an *ALCHIQ* knowledge base by using only *ABox* statements of the form $C(\mathbf{a})$ and $r(\mathbf{a}, \mathbf{b})$.

6 Modeling with DLs

*While frowning on plurality,
The pope likes cardinality:
It can enforce infinity,
And hence endorse divinity.
But, theologically speaking,
The papal theory needs tweaking
For it demands divine assistance
to prove “the three are one”-consistence.*

In this section, we will discuss the added value brought about by certain DL modeling features. We will also discuss specific types of statements for which some formalisms provide dedicated modeling primitives, although they are just “syntactic sugar,” that is they can be expressed by virtue of the modeling features already introduced. Moreover, we will provide some insight about model-theoretic consequences that arise from using or not using certain constructs.

Remark 37. Thereby, one can see that the expressive power of a logic can be characterized by its capability to “distinguish” interpretations. That is, a “stronger” logic might be able to distinguish two interpretations \mathcal{I}_1 and \mathcal{I}_2 meaning that there is a knowledge base \mathcal{KB} such that $\mathcal{I}_1 \models \mathcal{KB}$ but $\mathcal{I}_2 \not\models \mathcal{KB}$ (or vice versa), whereas a “weaker” logic may not have this capability. In many cases, this indistinguishability can be cast into statements of the following type: given any knowledge base \mathcal{KB} in a certain DL and a (set of) model(s) of \mathcal{KB} then performing a certain operation or manipulation on that model(s) will inevitably result in an interpretation which is again a model of \mathcal{KB} . We then say the set of models of \mathcal{KB} is *closed under* the considered operation.

6.1 A Lot Can Be Done in \mathcal{ALC}

Already \mathcal{ALC} features many modeling capabilities usually found in knowledge representation languages. Beyond the ones explicitly introduced, quite some more correspondences can be expressed indirectly. We will tackle the most important ones.

Concept Disjointness. Two concepts C and D are disjoint with respect to an interpretation \mathcal{I} , if their extensions do not overlap, i.e. $C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$. It is straightforward that this semantic condition can be cast into the GCI $C \sqcap D \sqsubseteq \perp$. Equivalently, this can be expressed by $C \sqsubseteq \neg D$ or $D \sqsubseteq \neg C$. Disjointness information is often neglected when doing logical modeling. It can, however, be very useful to derive negative information, e.g., the guarantee that some individual is *not* an instance of a concept.

Domain and Range of Roles. Given a role r , we may want to make statements about the source and target individuals for the respective relation. We say that the role r has *domain* C in an interpretation \mathcal{I} if any source individual of the relation associated with r is an instance of C , in other words, for every $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$, we have $\delta \in C^{\mathcal{I}}$. Likewise, we say that r has *target* D if for every

$\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$, also $\delta' \in D^{\mathcal{I}}$ is satisfied. The standard DLs covered here do not provide modeling primitives for specifying domain or range of a role, but they can be easily expressed with the means already present in \mathcal{ALC} . The above domain statement is equivalent to the GCI $\exists r.\top \sqsubseteq C$ whereas the range statement can be written as $\top \sqsubseteq \forall r.D$.

The Empty Role. It might seem a bit peculiar that, while \mathcal{SROIQ} supports both the universal and the empty concept (\top and \perp , respectively), it features only the universal role u whereas the empty role is not part of the definition. This is, however, not a severe omission as the empty role can be easily axiomatized: for a new role name `emptyRole` we can use the GCI $\top \sqsubseteq \forall \text{emptyRole}.\perp$ to force the extension of `emptyRole` to be empty. An alternative axiom (beyond \mathcal{ALC}) with the same effect is $\text{Dis}(u, \text{emptyRole})$.

Exercise 22. *Come up with an \mathcal{ALC} GCI that expresses the following statement: “If an academic supervises a project, then he is a project leader and the project is a research project.” Use the role name `supervises` as well as the concept names `Academic`, `Project`, `ProjectLeader`, and `ResearchProject`.*

6.2 Looking Back: Inverse Roles

Inverses allow for traversing roles in reverse direction. While DLs without inverses only allow for describing domain individuals by means of their “outgoing” roles, by means of inverses, “incoming” roles can be taken into account as well.

Example 38. Consider the interpretation \mathcal{I} from Example 16. It is rather easy to see that the domain individuals 3 and 5 (as well as any other prime number) are not distinguishable by \mathcal{ALC} concepts (in fact, not even by \mathcal{SROQ} concepts), that is, there is no concept C having 3 as an instance but not 5, or vice versa. On the other hand, the \mathcal{ALCI} concept $\exists \text{succ}^-. \exists \text{succ}^-. \exists \text{succ}^-. \neg \exists \text{succ}^-. \top$ does the job.

Moreover, some rather natural properties of relations can be expressed by means of inverses. A role r is called *symmetric* if for any $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$ also $\langle \delta', \delta \rangle \in r^{\mathcal{I}}$ holds, that is, relatedness via r always holds both ways. On the other hand it is called *asymmetric* if for all $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$ satisfy $\langle \delta', \delta \rangle \notin r^{\mathcal{I}}$ holds, this means that r -relatedness never holds both ways. Sometimes, symmetry or asymmetry of a role r is included in a DL as a separate axiom type, denoted by $\text{Sym}(r)$ or $\text{Asy}(r)$, respectively. The former can be easily expressed by stating that r has its own inverse as a subrole: $r^- \sqsubseteq r$. The latter can be characterized by stating that r and its inverse are disjoint: $\text{Dis}(r, r^-)$.

6.3 Model Manipulation Part I: Filtration

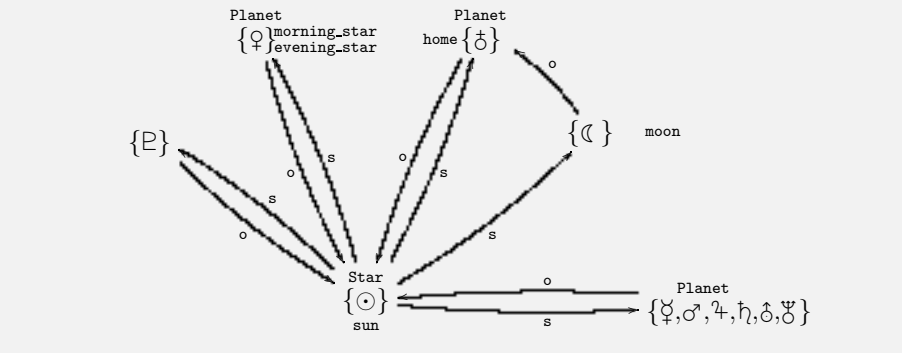
Now we will turn our attention to our first model transformation. Given a set \mathcal{C} of concepts and an interpretation \mathcal{I} , we can obtain the *filtration* of \mathcal{I} with respect

to \mathfrak{C} as follows: First, we define an equivalence relation \simeq on the domain elements of \mathcal{I} by letting $\delta \simeq \delta'$ for anonymous $\delta, \delta' \in \Delta^{\mathcal{I}}$ whenever δ and δ' coincide in terms of concept memberships for concepts from \mathfrak{C} , that is, for every $C \in \mathfrak{C}$ we have $\delta \in C^{\mathcal{I}}$ exactly if $\delta' \in C^{\mathcal{I}}$. Then, for some $\delta \in \Delta^{\mathcal{I}}$ we let $[\delta]_{\simeq} = \{\delta' \mid \delta \simeq \delta'\}$ and $\Delta^{\mathcal{I}}_{/\simeq} = \{[\delta]_{\simeq} \mid \delta \in \Delta^{\mathcal{I}}\}$. Verbally, the set $\Delta^{\mathcal{I}}_{/\simeq}$ consists of “bags” of domain elements from \mathcal{I} where all elements in one bag coincide on the concepts from \mathfrak{C} they satisfy. The filtration of \mathcal{I} is the interpretation \mathcal{J} with

- $\Delta^{\mathcal{J}} = \Delta^{\mathcal{I}}_{/\simeq}$
- for each $\mathbf{a} \in \mathbf{N}_I$, set $\mathbf{a}^{\mathcal{J}} = [\mathbf{a}^{\mathcal{I}}]_{\simeq}$;
- for each concept name $\mathbf{A} \in \mathbf{N}_C$, set $\mathbf{A}^{\mathcal{J}} = \{[\delta]_{\simeq} \mid \delta \in \mathbf{A}^{\mathcal{I}}\}$;
- for each role name $\mathbf{r} \in \mathbf{N}_R$, set $\mathbf{r}^{\mathcal{J}} = \{ \langle [\delta]_{\simeq}, [\delta']_{\simeq} \rangle \mid \langle \delta, \delta' \rangle \in \mathbf{r}^{\mathcal{I}} \}$;

Intuitively, this means, that the filtration is obtained by collapsing domain elements which are not distinguishable by virtue of concepts from \mathfrak{C} (nor by individual names) into one.

Example 39. Let \mathcal{I} be the interpretation from Example 14 and let \mathfrak{C} contain all \mathcal{ALC} concepts. Then the according filtration can be sketched as follows.



If, for a given \mathcal{SROI} knowledge base \mathcal{KB} , we let \mathfrak{C} be all concepts occurring in \mathcal{KB} (including the subexpressions of concepts) then the filtration of a model of \mathcal{KB} will again be a model of \mathcal{KB} . On the other hand, since in this case, \mathfrak{C} is finite, there can be only finitely many “bags” in $\Delta^{\mathcal{I}}_{/\simeq}$ which means that the filtration will even be a finite model of \mathcal{KB} . This allows to conclude that every satisfiable \mathcal{SROI} knowledge base has a finite model.

Remark 40. In general, logics for which the existence of an arbitrary model implies the existence of a model where $\Delta^{\mathcal{I}}$ is a finite set (usually briefly called *finite model*) are said to have the *finite model property*. This is a rather convenient property, since one may disregard infinite representations when looking for models of a knowledge base. Moreover, for any logic that has the finite model property and that can be embedded into FOL, the problem of knowledge base satisfiability is decidable.

Concluding, we can state that filtrations are quite stable in terms of modelhood preservation, however they fail as soon as cardinality constraints come into play.

Exercise 23. Consider Example 39 and find an \mathcal{ALCQ} axiom which is not satisfied in the interpretation given there although it is satisfied in the original interpretation from Example 14.

6.4 Up to Infinity: Cardinality Constraints

By means of cardinality constraints, precise statements about the number of individuals related to a certain individual via a role can be made. This kind of modeling features is of obvious practical value and wide-spread in other knowledge specification formalisms such as entity-relationship modeling or UML. Cardinality constraints also naturally capture certain role characteristics.

For instance, *role functionality* can be seen and treated as a special case of cardinality constraints. In words, a role is functional, if every domain individual is connected to at most one domain individual via the relation associated to that role. Formally, a role r is functional, if for every domain individual $\delta \in \Delta^{\mathcal{I}}$ there is at most one individual $\delta' \in \Delta^{\mathcal{I}}$ satisfying $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$. This condition can be enforced by the axiom $\top \sqsubseteq \leq 1.\top$. Sometimes, in DLs which do not support number restrictions in general, the according axiom is noted as $\text{Fun}(r)$. Typical examples for functional roles are `hasFather`, `marriedWith`, or `locatedInCountry`.

Remark 41. Note that by definition, a role can be functional and still not start from every domain individual, as in the case of `marriedWith`. Thus the term “functional” may be misleading as it may cause the erroneous impression that the role extension is a (total) function. Rather, functional roles semantically correspond to partial functions.

In fact, in the presence of cardinality constraints allows to enforce that a knowledge base has only models the domain of which is infinite. Consider the following knowledge base:

$$(\forall \text{succ}^-. \top)(\text{zero}) \quad \top \sqsubseteq \exists \text{succ}.\top \quad \top \sqsubseteq \leq 1.\text{succ}^-. \top$$

It is not to difficult to find a model for this knowledge base which has an infinite domain: in fact the interpretation described in Example 16 is such a model. On the other hand the knowledge base cannot have a model with finite domain.

Exercise 24. Prove this. Hint: assume a finite number of domain elements and count sources and targets for `succ`.

Note that we have just shown that any extension of \mathcal{ALCIF} does not have the finite model property.

6.5 Model Manipulation Part II: Unraveling

However, another nice property still holds in the presence of number restrictions. Roughly speaking, this property states that we can take an arbitrary model and “unfold” or “unroll” it such that all the parts of the model not containing named individuals are tree-like (i.e., cycle-free). More formally, the *unraveling* of an interpretation \mathcal{I} is an interpretation that is obtained from \mathcal{I} as follows: First, we define the set $S \subseteq (\Delta^{\mathcal{I}})^*$ of *paths* to be the smallest set of sequences of domain elements such that

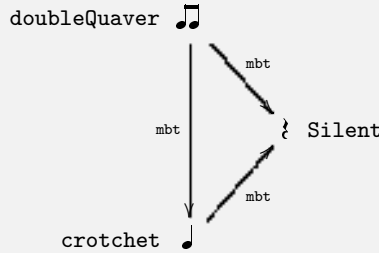
- for every $\mathbf{a} \in \mathbf{N}_I$, $\mathbf{a}^{\mathcal{I}}$ is a path;
- $\delta_1 \cdots \delta_n \cdot \delta_{n+1}$ is a path, if
 - $\delta_2 \neq \mathbf{a}^{\mathcal{I}}$ for all $\mathbf{a} \in \mathbf{N}_I$,
 - $\delta_1 \cdots \delta_n$ is a path,
 - $\delta_{i+1} \neq \delta_{i-1}$ for all $i = 2, \dots, n$,
 - $\langle \delta_n, \delta_{n+1} \rangle \in r^{\mathcal{I}}$ for some $r \in \mathbf{R}$.

For each $w = \delta_1 \cdots \delta_n \in S$, set $\text{last}(w) = \delta_n$. Now, we define the unraveling of \mathcal{I} as the interpretation $\mathcal{J} = \langle \Delta^{\mathcal{J}}, \cdot^{\mathcal{J}} \rangle$ with $\Delta^{\mathcal{J}} = S$ and, for each sequence $w \in \Delta^{\mathcal{J}}$, we define the interpretation of concept and role names as follows:

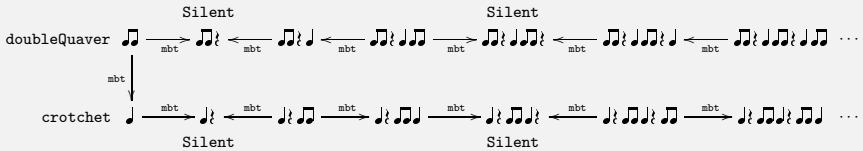
- (a) for each $\mathbf{a} \in \mathbf{N}_I$, set $\mathbf{a}^{\mathcal{J}} = \mathbf{a}^{\mathcal{I}}$;
- (b) for each concept name $\mathbf{A} \in \mathbf{N}_C$, set $w \in \mathbf{A}^{\mathcal{J}}$ iff $\text{last}(w) \in \mathbf{A}^{\mathcal{I}}$;
- (c) for each role name $\mathbf{r} \in \mathbf{N}_R$, set $\langle w, w' \rangle \in \mathbf{r}^{\mathcal{J}}$ iff
 - $w' = w\delta$ for some $\delta \in \Delta^{\mathcal{I}}$ and $\langle \text{last}(w), \delta \rangle \in \mathbf{r}^{\mathcal{I}}$ or
 - $w = w'\delta$ for some $\delta \in \Delta^{\mathcal{I}}$ and $\langle \delta, \text{last}(w') \rangle \in \mathbf{r}^{\mathcal{I}}$ or
 - $w = \mathbf{a}^{\mathcal{I}}$, $w' = \mathbf{b}^{\mathcal{I}}$ for some $\mathbf{a}, \mathbf{b} \in \mathbf{N}_I$ and $\langle \mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}} \rangle \in \mathbf{r}^{\mathcal{I}}$.

With this notion of unraveling we find that for any *ALCHIQ* knowledge base \mathcal{KB} , an interpretation \mathcal{I} is a model exactly if its unraveling is. This correspondence has some practical consequences: First it guarantees that *ALCHIQ* has the *forest model property*. That means that every satisfiable *ALCHIQ* knowledge base \mathcal{KB} has a model with a particular shape: there is a “root tangle” of named elements from which trees of anonymous elements grow. This property is for instance of particular interest to prove the completeness of tableau algorithms.

Example 42. To demonstrate what happens during the unraveling of an interpretation, consider this small example interpretation (where `mbt` is intended to mean “more beats than”):



In order to unravel this interpretation, intuitively, we first pick all named individuals (i.e., ♩ and ♪) and keep them as well as their mutual relationships. Then, in the original interpretation, we walk along the (incoming and outgoing) role links to anonymous elements to find the named individuals’ role neighbors, these neighbors are (as well as the corresponding role links) reproduced in the unraveling. Even if the neighbors are the same, we introduce separate copies in the unraveling, using the “origin element” as a prefix to distinguish them. In our example, we introduce $\text{♩}?$ as the `mbt`-neighbor of ♩ (caused by $?$ in the original interpretation) and $\text{♪}?$ as the `mbt`-neighbor of ♪ (caused by the same $?$). We then proceed to neighbors of neighbors and so forth. Thereby, we exclude the elements that we “just came from” in the previous step. We may, however, traverse elements of the original interpretation several times, we will however disregard their names and create anonymous copies of them in the unraveling. In our case, the result of this procedure is an infinite interpretation which is partially depicted below.



Exercise 25. Sketch or formally describe the unravelings of the interpretations from Example 14 and Example 16. For the latter and for Example 42, give one axiom from the DL \mathcal{S} and one from the DL \mathcal{ALCCOI} , either of which hold in the interpretation but not the according unraveling.

Remark 43. In fact, variants of the forest model property also hold for some DLs containing role chain axioms and/or nominal concepts, requiring also to modify the employed unraveling technique. In the presence of role chain axioms, one usually defines a “skeleton” of the model via unraveling into a forest structure and thereafter adds further “role links” the presence of which is enforced by the RIAs. In the presence of nominals, one has to allow so-called “backlinks” i.e. tree individuals are allowed to have role links back into the root tangle (but not into other trees).

6.6 Far Far Away: Transitivity

Transitivity of a role r is expressible by the complex role inclusion $r \circ r \sqsubseteq r$. In DLs that do not feature any complex role inclusions but transitivity this axiom is often alternatively written as $\text{Tra}(r)$. Role transitivity statements come about quite naturally for a variety of relations that are to be modeled. Typical examples for transitive roles are `ancestorOf`, `superiorOf`, `partOf`, `greaterThan`, etc. Role transitivity declarations allow for a more succinct modeling and better querying capabilities via entailment checks.

Example 44. Envisioning a company and a knowledge base containing employee data, it would of course be possible to explicitly add all superior relations as ABox role assertions `superiorOf(a, b)`. On the other hand, the same can be achieved (in terms of inferrable superior information) by only adding role assertions for the cases of where \mathbf{a} denotes a direct superior of \mathbf{b} , if we additionally state that `superiorOf` is transitive. Moreover this second version is advantageous in terms of maintenance: whenever a new employee joins the company, only their direct superior(s) and inferior(s) need to be explicitly specified.

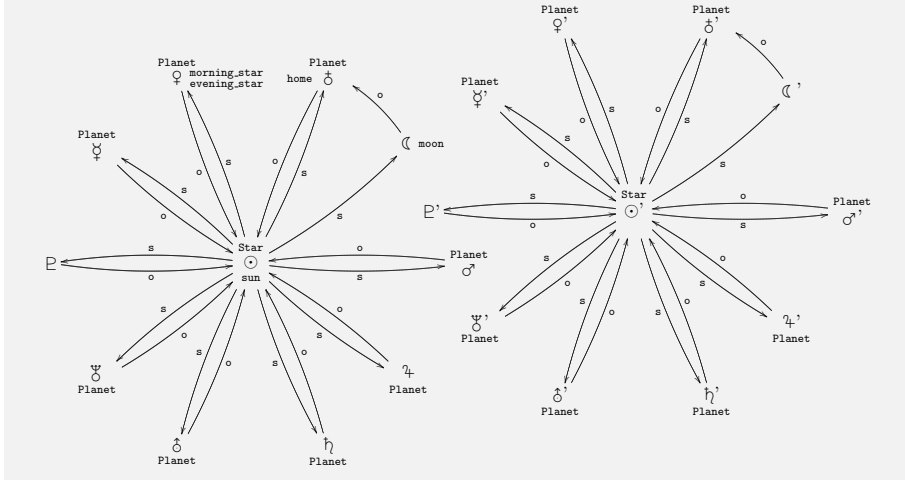
However, what can be expressed in terms of transitivity in standard DLs is limited. Thereby the limitations are inherited from FOL. What cannot be done in the DLs treated here is to precisely talk about the transitive closure of a given role. In other words, there is no way to axiomatize the condition that one role r is the transitive closure of another role s (formally, this condition can be expressed by $r^{\mathcal{I}} = (s^{\mathcal{I}})^*$). What can be done is to say that the extension of r contains the transitive closure of s (i.e. $(s^{\mathcal{I}})^* \subseteq r^{\mathcal{I}}$) by specifying $s \sqsubseteq r$ and $r \circ r \sqsubseteq r$. Presuming this axiomatization of an upper bound for the transitive closure, we can e.g. check whether there is an “ s -path” of arbitrary length from an individual \mathbf{a} to an individual \mathbf{b} in every model of the knowledge base by checking whether the knowledge base entails the role assertion $s(\mathbf{a}, \mathbf{b})$. Still, there is no way to check for the necessary absence of such a path in all models of the knowledge base.

6.7 Model Manipulation Part III: Disjoint Union

We now consider a transformation which, roughly speaking, takes two interpretations and puts them side by side. More formally, given two interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ and $\mathcal{J} = (\Delta^{\mathcal{J}}, \cdot^{\mathcal{J}})$, assuming that $\Delta^{\mathcal{I}} \cap \Delta^{\mathcal{J}} = \emptyset$, we define the *disjoint union* of \mathcal{I} with \mathcal{J} denoted by $\mathcal{I}+\mathcal{J} = (\Delta^{\mathcal{I}+\mathcal{J}}, \cdot^{\mathcal{I}+\mathcal{J}})$ as follows: $\Delta^{\mathcal{I}+\mathcal{J}} = \Delta^{\mathcal{I}} \cup \Delta^{\mathcal{J}}$, $\mathbf{a}^{\mathcal{I}+\mathcal{J}} = \mathbf{a}^{\mathcal{I}}$, $\mathbf{A}^{\mathcal{I}+\mathcal{J}} = \mathbf{A}^{\mathcal{I}} \cup \mathbf{A}^{\mathcal{J}}$ and $\mathbf{r}^{\mathcal{I}+\mathcal{J}} = \mathbf{r}^{\mathcal{I}} \cup \mathbf{r}^{\mathcal{J}}$. Note that, unlike most definitions of disjoint unions, this definition is asymmetric since, for the mapping of the individuals, preference is given to \mathcal{I} . One can show that whenever \mathcal{I} and \mathcal{J} are models of a *SHIQ* knowledge base \mathcal{KB} then so is their disjoint union $\mathcal{I}+\mathcal{J}$.

Exercise 26. Prove the claim above. Hint: An intermediate lemma showing $C^{\mathcal{I}+\mathcal{J}} = C^{\mathcal{I}} \cup C^{\mathcal{J}}$ will come handy for that. This will require a structural induction over the concepts.

Example 45. Given the interpretation \mathcal{I} from Example 14, let \mathcal{I}' denote \mathcal{I} where every domain element δ has been renamed into δ' . Then the interpretation $\mathcal{I} + \mathcal{I}'$ can be displayed as follows:



In fact, the above result can be generalized to disjoint unions of infinitely many models. This gives rise to a property which could be called the *infinite model property*: whenever there is an arbitrary model for a *SHIQ* knowledge base \mathcal{KB} , then there is also an infinite one.

Remark 46. More generally, these properties even hold for all *SRIQ* knowledge bases not containing the universal role.

Exercise 27. Consider Example 45 and find an *ALCO* axiom which is not satisfied in the interpretation given there although it is satisfied in the original interpretation from Example 14.

Wrapping up, what we have learned about model manipulations, their range of applicability, and the model properties they give rise to can be summarized in the following table.

manipulation	preserves models for	associated property
filtration	<i>SROI</i>	finite model property
unraveling	<i>ALCHIQ</i>	forest model property
disjoint union	<i>SRIQ</i> \ u	infinite model property

6.8 Know Your Bounds: Nominal Concept and Universal Role

The modeling power brought about by nominal concepts and universal roles is quite similar. For instance, having the universal role at disposal, we can remove all nominal concepts from a *SROIQ* knowledge base as follows: first,

rewrite every nominal concept $\{\mathbf{a}_1, \dots, \mathbf{a}_n\}$ into $\{\mathbf{a}_1\} \sqcup \dots \sqcup \{\mathbf{a}_n\}$ according to the equivalence given in Section 5. Next, introduce fresh concept names $\mathbf{A}_{\{\mathbf{a}\}}$ for all singleton nominal concepts thus obtained and substitute every occurrence of any $\{\mathbf{a}\}$ by the according $\mathbf{A}_{\{\mathbf{a}\}}$. Finally, add the concept assertion $\mathbf{A}(\mathbf{a})$ as well as the GCI $\top \sqsubseteq \leq 1u.\mathbf{A}_{\{\mathbf{a}\}}$ for any introduced $\mathbf{A}_{\{\mathbf{a}\}}$.

On the other hand, the universal role can be emulated once nominal concepts are allowed: we introduce a fresh individual name **center** and a new role name **toCenter** and force every individual to have a **toCenter** relation to the individual denoted by **center** by means of the axiom $\top \sqsubseteq \exists \text{toCenter}.\{\mathbf{center}\}$. Now we can get from every domain individual to every other by a two-hop travel along **toCenter** and **toCenter**⁻. Thus we can replace every $\exists u.C$ with $\exists \in \{\forall, \exists, \leq n, \geq n\}$ by the concept expression $\exists \text{toCenter}.\exists \text{toCenter}^-.C$.

Exercise 28. Find a way to remove the *RBox* occurrences of *u* as well.

A crucial feature showing the added expressivity obtained from nominal concepts or the universal role is the capability to bound or fix the number of individuals in the extension of a class or even in the whole domain. Both the GCIs $\text{AtMostTwo} \sqsubseteq \{\mathbf{one}, \mathbf{two}\}$ and $\top \sqsubseteq \leq 2u.\text{AtMostTwo}$ specify that the concept **AtMostTwo** has at most two instances in every model. In order to cause the extension size to be exactly two, we would have to add $\mathbf{one} \not\approx \mathbf{two}$ or $\top \sqsubseteq \geq 2u.\text{AtMostTwo}$, respectively. Likewise, we can enforce the whole domain to contain at most (or exactly) two individuals by imposing these axiom with **AtMostTwo** substituted by \top .

Remark 47. These considerations show that as soon as nominal concepts or the universal role is involved, models of knowledge bases need not be closed under disjoint union as it was the case for e.g. *SHIQ*.

Exercise 29. As we have seen, *SROIQ* allows to enforce that the domain size (i.e. the number of its elements) is at most n for any given $n \in \mathbb{N}$. Contemplate whether there is a knowledge base $\mathcal{KB}_{\text{fin}}$ that emulates finite models, i.e., for every knowledge base \mathcal{KB} not using vocabulary from $\mathcal{KB}_{\text{fin}}$ the models of $\mathcal{KB} \cup \mathcal{KB}_{\text{fin}}$ are exactly those models of \mathcal{KB} with finite domain, if one abstracts from the vocabulary of $\mathcal{KB}_{\text{fin}}$.

Exercise 30. Is it possible to create a *SHIQ* knowledge base \mathcal{KB} such that every model contains one individual which is connected via a role **r** to infinitely many other individuals? Can the same be achieved in *ALCHOIQ*? What about *ALCHIQ*? For each of the cases either provide such a knowledge base or argue why this is not possible.

6.9 Selfishness

The self concept enables to speak about “role loops”, i.e. situations where an individual is simultaneously source and target of the same relation, or in other

words the individual is connected to *itself*. This allows to define concepts based on such situations, for instance we could define `PersonCommittingSuicide` $\equiv \exists \text{kills.Self}$ or `Narcissist` $\equiv \exists \text{loves.Self}$. Beyond that, this feature comes handy when global properties of roles are to be enforced. A role r is said to be *reflexive* if its associated relation is, i.e. if $\langle \delta, \delta \rangle \in r^{\mathcal{I}}$ for all $\delta \in \Delta^{\mathcal{I}}$. Conversely, it is called *irreflexive* if $\delta \neq \delta'$ for all $\langle \delta, \delta' \rangle \in r^{\mathcal{I}}$. In some places, the definition of *SRONTQ* includes additional RBox axioms of the form $\text{Ref}(r)$ or $\text{Irr}(r)$ to specify reflexivity or irreflexivity of r , respectively. However, these role characteristics can be equivalently expressed by the GCIs $\top \sqsubseteq \exists r.\text{Self}$ or $\exists r.\text{Self} \sqsubseteq \perp$, respectively.

Exercise 31. *If one has a closer look into the literature, these additional axiom types require r to be simple in the case of irreflexivity but not in the case of reflexivity statements. In our current translation, role simplicity would be required in both cases. How can this restriction be circumvented by an alternative translation of the reflexivity statement?*

6.10 Open World vs. Closed World

A useful distinction often made in the context of logic-based information systems is that between *closed-world* and *open-world* reasoning. Essentially, this distinction is concerned with the question how missing information is treated. Under the *closed-world assumption* (CWA) facts which cannot be deduced from a knowledge base are supposed to be *false* whereas under the *open-world assumption* the truth of these facts is simply *unknown*. Expert or database systems often implement the CWA. Opposed to this, as a consequence of the semantics introduced in Section 3, DLs follow the OWA. This is also implied by the fact, that (most) DLs are fragments of first-order logic, which also adheres to the OWA.

Example 48. Consider the following knowledge base \mathcal{KB} containing merely ABox statements:

<code>Planet(home)</code>	<code>orbitsAround(home, sun)</code>
<code>Planet(morning_star)</code>	<code>orbitsAround(moon, home)</code>
<code>Star(sun)</code>	<code>orbitsAround(morning_star, sun)</code>
<code>evening_star \approx morning_star</code>	

While `Planet(evening_star)` and `orbitsAround(evening_star, sun)` are consequences of \mathcal{KB} , negated statements like $\neg \text{Star}(\text{home})$ or $\neg \text{orbitsAround}(\text{sun}, \text{moon})$ or `moon $\not\approx$ home` are not due to the OWA. This can be explained by the fact, that there are models for the \mathcal{KB} where these statements do not hold (but rather their unnegated variants). In order to enforce these negated statements they would have to be explicitly added to the knowledge base.

While the OWA is commonly argued to be the right perspective in the context of the Semantic Web where completeness seems to be hard to achieve, there are cases, where e.g. the extension of a concept or a role is entirely known and one

wants to express this information in order to guarantee that the according additional consequences can be drawn. To a certain extent, this can be implemented by virtue of nominal concepts.

Example 49. Revisiting Example 48, to obtain the consequence $\neg\text{Star}(\text{home})$ we could alternatively state that `sun` is the name of the only individual belonging to the concept `Star` by adding the TBox axiom $\text{Star} \sqsubseteq \{\text{sun}\}$. This has the advantage that also the concept membership of anonymous individuals is thereby excluded which cannot be achieved by ABox statements. Yet, in order to get the above consequence we still have to additionally assert $\text{sun} \not\approx \text{home}$, thereby excluding the case that `home` and `sun` refer to the same individual. In the same way, we can treat roles. For example, the axiom $\{\text{home}\} \sqsubseteq \forall \text{orbitsAround}.\{\text{sun}\}$ expresses that `home` is `orbitsAround`-connected to nothing but (possibly) `sun`.

While nominals come handy for making “nothing but” statements, they cannot fully simulate closed-world behavior. Therefore (local) closed-world extensions to DLs have been investigated. Notable approaches in that direction are (*auto*)*epistemic DLs*, and *circumscriptive DLs*.

7 Reasoning Tasks and Their Reducibility

*A knowledge base with statements in it
Seeks a model sound and nice
No matter, finite or infinite,
It asks a hermit for advice.
Yet, shattering is the reaction:
“Inconsistency detection,
You can’t get no satisfaction.”*

It is one of the major selling points of logic-based knowledge representation in general and of DLs in particular that, once a body of knowledge has been accumulated and transferred into a logical representation, this knowledge can be queried and worked with in an intelligent way which goes well beyond what can be done with traditional information systems such as databases. In this section we will review typical tasks that can be performed with DL knowledge bases and that require elaborate inferencing. We can see that some of those tasks can be reduced to others which alleviates the task of creating tools performing those tasks.

7.1 Knowledge Base Satisfiability

Remember that a knowledge base \mathcal{KB} is called satisfiable (also: consistent) if it has a model, i.e., there is an interpretation \mathcal{I} with $\mathcal{I} \models \mathcal{KB}$, otherwise it is called unsatisfiable, inconsistent, or contradictory. Deciding whether a knowledge base is consistent is important in its own right, as knowledge base inconsistency often hints at severe modeling errors: since knowledge bases are supposed to describe real state of affairs, they should not be contradictory. Moreover, due to the principle of explosion, an inconsistent knowledge base entails every statement

which renders any derived information useless. Additionally, as we will see in a bit, axiom entailment checks can be reduced to detecting inconsistency of knowledge bases.

7.2 Axiom Entailment

We remember that a knowledge base \mathcal{KB} entails a DL axiom α if every model of \mathcal{KB} is also a model of α . Axiom entailment can be seen as the prototypical reasoning task for querying knowledge: given a body of knowledge formally specified in a knowledge base, this knowledge is to be “logically queried” by checking whether some statement is necessarily true, presuming the statements of the knowledge base.

The problem of checking axiom entailment can be reduced to deciding knowledge base satisfiability. The idea behind this reduction is proof by contradiction: we show that something holds by assuming the opposite and deriving a contradiction from that assumption. Suppose α and β are axioms claiming the opposite of each other. Then every interpretation (hence in particular every model of the knowledge base \mathcal{KB}) satisfies either α or β , but not both. Now, if α is a consequence of \mathcal{KB} , we know that every model of \mathcal{KB} is a model of α . This means that no model of \mathcal{KB} can be a model of β . In other words, the extended knowledge base $\mathcal{KB}' = \mathcal{KB} \cup \{\beta\}$ can have no model which just means that \mathcal{KB}' is unsatisfiable. Thus the axiom entailment problem can be easily recast into a knowledge base unsatisfiability problem, provided we find such an “opposite” axiom for the given α . In *SROIQ* this is obvious for some cases. In some other cases, we have to revert to finding an axiom or a set of axioms emulating the opposite of α , which works just as well. We give the correspondences for all types of *SROIQ* axioms in Table 1.

Table 1. Definition of axiom sets \mathcal{A}_α such that $\mathcal{KB} \models \alpha$ exactly if $\mathcal{KB} \cup \mathcal{A}_\alpha$ is unsatisfiable. Individual names c with possible subscripts are supposed to be fresh. For GCIs (third line), the first variant is normally employed, however, we also give a variant which is equivalent instead of just emulating.

α	\mathcal{A}_α
$r_1 \circ \dots \circ r_n \sqsubseteq r$	$\{\neg r(c_0, c_n), r_1(c_0, c_1), \dots, r_n(c_{n-1}, c_n)\}$
$\text{Dis}(r, r')$	$\{r(c_1, c_2), r'(c_1, c_2)\}$
$C \sqsubseteq D$	$\{(C \sqcap \neg D)(c)\}$ or: $\{\top \sqsubseteq \exists u(C \sqcap \neg D)\}$
$C(a)$	$\{\neg C(a)\}$
$r(a, b)$	$\{\neg r(a, b)\}$
$\neg r(a, b)$	$\{r(a, b)\}$
$a \approx b$	$\{a \not\approx b\}$
$a \not\approx b$	$\{a \approx b\}$

7.3 Concept Satisfiability

Given a knowledge base \mathcal{KB} , a concept $C \in \mathbf{C}$ is called *satisfiable* with respect to \mathcal{KB} , if it may contain individuals, i.e. there is a model \mathcal{I} of \mathcal{KB} that maps C to a nonempty set, formally: $C^{\mathcal{I}} \neq \emptyset$. Obviously, there are concepts which are unsatisfiable irrespective of the underlying knowledge base, like $\mathbf{A} \sqcap \neg \mathbf{A}$ or simply \perp . If, however some atomic concept $\mathbf{A} \in N_I$ is unsatisfiable, this may as well indicate modeling errors. A knowledge base where all atomic concepts are satisfiable is usually called *coherent*. Note that a knowledge base can be incoherent but satisfiable. Like knowledge base satisfiability and axiom entailment, concept satisfiability is a decision problem, i.e. we get *yes* or *no* as an answer.

The problem of deciding concept satisfiability can be reduced to axiom entailment. An unsatisfiable concept C is necessarily empty in any model \mathcal{I} , i.e., $C^{\mathcal{I}} = \emptyset$. This can be rewritten into $C^{\mathcal{I}} \subseteq \emptyset$ (since the other direction is trivial), and further (using the fact that $\perp^{\mathcal{I}} = \emptyset$) into $C^{\mathcal{I}} \subseteq \perp^{\mathcal{I}}$. However this means $\mathcal{I} \models C \sqsubseteq \perp$ for every model \mathcal{I} of \mathcal{KB} , therefore $\mathcal{KB} \models C \sqsubseteq \perp$. Hence, unsatisfiability of a concept C with respect to some knowledge base \mathcal{KB} can be decided by checking whether \mathcal{KB} entails the GCI $C \sqsubseteq \perp$.

7.4 Instance Retrieval

Given a knowledge base \mathcal{KB} and a concept C , it is a rather natural desire to ask for C 's instances. However, there are two issues with that: First, a knowledge base usually has many models and a specific individual may be instance of C in one model but not in another. So, one typically asks for individuals which are instances of C in *every* model. The other problem is that from model to model, the domain $\Delta^{\mathcal{I}}$ may vary and does not need to contain the same individual. The only way to refer to individuals in a sensible, cross-domain way is via their names. This is why one restricts to named individuals for the instance retrieval task. Consequently, the task could be formulated as follows: given a knowledge base \mathcal{KB} and a concept C , give me all individual names $\mathbf{a} \in N_I$ for which $\mathbf{a}^{\mathcal{I}} \in C^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{KB} .

Remark 50. This definition of instance retrieval may even lead to the peculiar case that one can infer from a knowledge base that a concept C is nonempty in every model (which can e.g. be tested by asking whether $\mathcal{KB} \models \top \sqsubseteq \exists u.C$) while the instance retrieval for C yields nothing. A simple example for this would be the knowledge base containing only the axiom $(\exists \mathbf{r}.C)(\mathbf{a})$.

Given the definition of instance retrieval above, it is obvious that an individual name \mathbf{a} will be delivered as part of the answer of an instance retrieval with respect to a concept C precisely if $\mathcal{KB} \models C(\mathbf{a})$. Therefore, instance retrieval can be performed by successively checking whether the considered knowledge base entails $C(\mathbf{a})$ for every individual name \mathbf{a} . This takes $|N_I(\mathcal{KB})|$ entailment checks. Depending on what concrete reasoning algorithm is employed, fewer calls to the reasoning procedure may be required since it might be possible to retrieve many

instances at once. This particularly applies to reasoning methods based on logic programming and/or database systems.

Sometimes, the term instance retrieval is also used for roles. In that case we are looking for all pairs $\langle \mathbf{a}, \mathbf{b} \rangle$ of individual names $\mathbf{a}, \mathbf{b} \in \mathbf{N}_I$ for which $\langle \mathbf{a}^{\mathcal{I}}, \mathbf{b}^{\mathcal{I}} \rangle \in r^{\mathcal{I}}$ for every model \mathcal{I} of \mathcal{KB} . This can be easily checked by asking for the entailment $\mathcal{KB} \models r(\mathbf{a}, \mathbf{b})$ for every combination of individual names.

7.5 Classification

Given a knowledge base \mathcal{KB} , the concept names occurring therein can be put into a hierarchy according to their subsumption relationships. More precisely, if we define a relation $\sqsubseteq_{\mathcal{KB}}$ on the set \mathbf{N}_C of concept names by $\mathbf{A} \sqsubseteq_{\mathcal{KB}} \mathbf{B}$ iff $\mathcal{KB} \models \mathbf{A} \sqsubseteq \mathbf{B}$, we find that this relation is a *preorder*, that is, we have $\mathbf{A} \sqsubseteq_{\mathcal{KB}} \mathbf{A}$ for all $\mathbf{A} \in \mathbf{N}_C$ and from $\mathbf{A} \sqsubseteq_{\mathcal{KB}} \mathbf{B}$ and $\mathbf{B} \sqsubseteq_{\mathcal{KB}} \mathbf{C}$ follows $\mathbf{A} \sqsubseteq_{\mathcal{KB}} \mathbf{C}$.

Exercise 32. Prove that $\sqsubseteq_{\mathcal{KB}}$ is indeed a preorder.

Classification of a knowledge base is the task of entirely determining $\sqsubseteq_{\mathcal{KB}}$. This task is practically important due to several reasons: During the knowledge base modeling process, the modeler has an overview over the hierarchical structure of the used concept names which can be diagrammatically visualized in a nice, intuitive way. On the other hand, classification can serve as a preprocessing step that speeds up subsequently performed reasoning tasks with respect to the underlying knowledge base.

Obviously, classification of a knowledge base can be performed by checking the entailment $\mathcal{KB} \models \mathbf{A} \sqsubseteq \mathbf{B}$ for any pair \mathbf{A}, \mathbf{B} of class names, which amounts to $|\mathbf{N}_C| \cdot (|\mathbf{N}_C| - 1)$ separate entailment checks. However, exploiting the properties of preorders and concept subsumption statements explicitly given by GCIs, the number of such checks can be drastically reduced [Shearer and Horrocks, 2009].

7.6 Conjunctive Query Answering

Conjunctive queries (CQs) and *unions of conjunctive queries* (UCQs) are well known in the database community [Chandra and Merlin, 1977] and constitute an expressive query language with capabilities that go well beyond standard reasoning tasks in DLs. In terms of first-order logic, these CQs and UCQs are formulae from the positive existential fragment. Free variables in a query (not bound by an existential quantifier) are also called *answer variables* or *distinguished variables*, whereas existentially quantified variables are called *non-distinguished*. As an example, $\exists y \exists z (\text{childOf}(x, y) \wedge \text{childOf}(x, z) \wedge \text{married}(y, z))$ with distinguished variable x and non-distinguished variables y and z represents a conjunctive query asking for all children whose parents are married with each other. If all variables in the query are non-distinguished, the query answer is just *true* or *false* and the query is called a *Boolean query*. Given a knowledge base \mathcal{KB} and a Boolean UCQ q , the query entailment problem is deciding whether q is *true* or *false* w.r.t. \mathcal{KB} , i.e., we have to decide whether each model of \mathcal{KB} provides for a suitable

assignment for the variables in q .² For a query with distinguished variables, the answers to the query are those tuples of individual names (constants) for which the knowledge base entails the query that is obtained by replacing the free variables with the individual names in the answer tuple. The problem of finding all answer tuples is known as query answering.

In general, conjunctive query answering or checking Boolean conjunctive query entailment are not easily (more precisely: polynomially) reducible to any of the other standard reasoning tasks treated above, which can be concluded from the fact that the worst-case complexities for these problems are usually way harder than the complexities of the other tasks [Lutz, 2008]. Conversely, it is trivial to reduce the task of checking knowledge base consistency to checking conjunctive query entailment: for instance, \mathcal{KB} is inconsistent exactly if for fresh concept names A_{aux} and B_{aux} the knowledge base $\mathcal{KB} \cup \{A_{aux} \sqcap B_{aux} \sqsubseteq \perp\}$ satisfies the conjunctive query $\exists x(A_{aux}(x) \wedge B_{aux}(x))$.

Exercise 33. A conjunctive query is called *tree-shaped* if for any two query variables x, y there is exactly one sequence of pairwise different query variables z_0, \dots, z_n and exactly one sequence r_1, \dots, r_n of role names such that $z_0 = x$, $z_n = y$, and for every $1 \leq i \leq n$ either $r_i(z_{i-1}, z_i) \in q$ or $r_i(z_i, z_{i-1}) \in q$. Argue that query answering for a tree-shaped conjunctive query with one distinguished variable can be reduced to (concept) instance retrieval.

7.7 Other Reasoning Tasks

The reasoning tasks described above, excluding conjunctive query answering, are often referred to as *standard reasoning tasks*. Still, conjunctive query answering is conceptually in line with those, since it can be formulated as entailment checking. Beyond those *deductive* tasks which are all concerned with determining logical consequences, there are several *non-standard reasoning tasks* where the goal is somewhat different. In the following, we will briefly go through a selection of these.

Induction. As opposed to the aforementioned deductive methods, inductive approaches³ usually take an amount of factual (assertional) data and try to generalize therefrom by generating hypotheses expressed as terminological axioms or complex concepts. This sort of reasoning tasks are related to data mining problems and respective approaches draw their inspiration from machine learning and in particular inductive logic programming (ILP, [Lehmann, 2009]). Since inductive reasoning is not truth-preserving (i.e. hypotheses which are generated may be falsified), also interactive methods with human expert involvement have been proposed [Rudolph, 2004].

² Note that in general, solving this task is way harder than querying a classical database, as the considered models may be infinite in both size and number.

³ Not to be confused with the mathematical proof technique of induction.

Abduction. Like induction and unlike deduction, abduction is an inferencing method which is not truth-preserving. Roughly speaking, abduction could be described as “premise guessing.” More precisely, given a knowledge base \mathcal{KB} in some DL and an axiom α such that α is not entailed by \mathcal{KB} , abductive reasoning is concerned with finding a knowledge base \mathcal{KB}' with specific properties such that α is a logical consequence of $\mathcal{KB} \cup \mathcal{KB}'$. In ontology engineering, abductive reasoning services come handy when a wanted consequence is not entailed and one wants to determine what information is missing [Noia *et al.*, 2009].

Explanation. If results of automated reasoning are to be shared with human users, it is often not sufficient to just display the result. Often it is also desirable to give an account on the cause why some axiom is entailed by the knowledge base, in other words to give an *explanation* for it. In most cases, only few axioms actually contribute to an entailment. Thus it is already quite helpful to find a minimal subset of a knowledge base for which the entailment still holds. More precisely, given a knowledge base \mathcal{KB} and an axiom α with $\mathcal{KB} \models \alpha$, a *justification* for the entailment is a knowledge base $\mathcal{KB}' \subseteq \mathcal{KB}$ such that $\mathcal{KB}' \models \alpha$ but for every $\mathcal{KB}'' \subset \mathcal{KB}'$ holds $\mathcal{KB}'' \not\models \alpha$. In general, a justification does not need to be unique, there might be more than one justification for an entailment [Horridge *et al.*, 2008].

Module extraction. When confronted with large knowledge bases, it might be worthwhile to identify natural partitions of them which logically interact with each other only in a restricted way, such that they can be handled independently when it comes to query answering or reasoning in general. In other cases, one may be interested only in a part of the knowledge specified in a knowledge base which is expressible in a certain fraction of the vocabulary. In general, the task of identifying or computing such knowledge base parts is referred to as *module extraction* [Stuckenschmidt *et al.*, 2009].

8 Algorithmic Approaches to DL Reasoning

*Is it consequence-driven
Automatically given
What we base our system upon?
Or do, fueled by Rousseau,
we say “Guerre aux tableaux!
Et vive la resolution!”?*

Various reasoning paradigms have been investigated with respect to their applicability to DLs. Most of them originate from well-known approaches for theorem proving in a first-order logic setting. However, in contrast to the unavoidable downside that reasoning methods for first-order logic cannot be sound, complete, and terminating, approaches to reasoning in DLs aim at a sound and complete decision procedures, whence the adopted reasoning techniques have to be adapted in order to guarantee termination.

The majority of state-of-the art OWL reasoners, such as Pellet [Sirin *et al.*, 2007], FaCT++ [Tsarkov and Horrocks, 2006], or RacerPro

[Haarslev and Möller, 2001] use tableau methods with good performance results, but even those successful systems are not applicable in all practical scenarios. This motivates the search for alternative reasoning approaches that employ different methods in order to address cases where tableau algorithms exhibit certain weaknesses. Successful examples in this respect are the works based on resolution [Motik and Sattler, 2006] and hyper-tableaux [Motik *et al.*, 2009c] as well as consequence-based approaches [Kazakov, 2009].

As we have seen in the previous section, many important reasoning tasks can be reduced to checking knowledge base satisfiability, hence we will focus on this specific task. In general, reasoning methods can be subdivided into *model-theoretic methods* on one hand and *proof-theoretic methods* on the other.

Model-theoretic methods essentially try to construct models of a given knowledge base in an organized way. If this succeeds, the knowledge base has obviously been shown to be satisfiable, if it can be shown that the construction must necessarily fail, unsatisfiability has been established. Typical reasoning paradigms of that sort are tableau procedures and automata-based approaches.

Remark 51. If models are represented explicitly (i.e., for an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ both $\Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}}$ are stored in some data structure), a naïve model construction strategy can only arrive at finite models, obviously. While this may be enough for logics that satisfy the finite model property, it is insufficient in the general case. However, this problem can be circumvented if one reverts to *finite model representations*, which only store a finite part of the model explicitly and provide additional (finite) information how this partial model could be deterministically extended into a real model. Intuitively, this can be compared to the decimal representation of rational numbers: while the correct value of $\frac{13}{11} = 1.18181818 \dots$ needs infinitely many digits to be precisely noted down, it is not hard to come up with a finite representation, namely $1.\overline{18}$ which, by virtue of the additional extra information provided by the overline, shows how the infinite “pure” representation could be constructed (if one had infinite time and memory). Of course, when working with finite representations, it is crucial that these allow for effective detection of axiom satisfaction.

As opposed to model-theoretic reasoning methods, proof-theoretic approaches operate more on the syntactic side: starting out from a normalized version of the knowledge base, deduction rules are applied to derive further logical statements about a potential model. If, in the course of these derivations an overt contradiction is derived, the considered knowledge base has shown to be unsatisfiable. In order to guarantee a termination of the procedure also in the case of satisfiability it is crucial that in the course of derivation, some sort of saturation will be reached in finite time. This can e.g. be achieved by restricting the relevant propositions (which may or may not be derived) to a finite set.

In the following, we will survey some well-known reasoning paradigms for DLs without going into technical details.

8.1 Tableau

Tableau procedures aim at constructing a model that satisfies all axioms of the given knowledge base. The strategy here is to maintain a set D of elements representing domain individuals (including anonymous ones) and acquire information about their concept memberships and role interrelatedness. D is initialized by all the individual names and the according ABox facts. Normally, the partial model thus constructed does not satisfy all the TBox and RBox axioms. Thus, the intermediate model is “repaired” as required by the axioms. This may mean to establish new concept membership or role interrelatedness information about the maintained elements, yet sometimes it may also be necessary to extend the set of considered domain individuals. Now and again, it might be required to make case distinctions and backtrack later. If we arrive at a state, where the intermediate model satisfies all the axioms and hence does not need to be repaired further, the knowledge base is satisfiable. If the intermediate model contains overt contradictions (such as an element marked as instance of a concept C and its negation $\neg C$ or an element marked as an instance of \perp), we can be sure that repairing it further by adding more information will never lead to a proper model, hence we are in a “dead end” need to backtrack. If every alternative branch thus followed leads into such a “dead end”, we can be sure that no model can exist.

Example 52. Omitting a lot of technical details, we shortly explain how the satisfiability of the knowledge base from Example 12 would be established by a tableau algorithm. For better reference, we first recap the knowledge base.






$$\text{owns} \sqsubseteq \text{caresFor} \quad (5)$$

$$\text{Healthy} \sqsubseteq \neg \text{Dead} \quad (6)$$

$$\text{Cat} \sqsubseteq \text{Dead} \sqcup \text{Alive} \quad (7)$$

$$\text{HappyCatOwner} \sqsubseteq \exists \text{owns.Cat} \sqcap \forall \text{caresFor.Healthy} \quad (8)$$

$$\text{HappyCatOwner}(\text{schrödinger}) \quad (9)$$

As explained we first initialize the set of domain elements by letting $D = \{\text{schrödinger}\}$, moreover, due to the only ABox axiom (9) we mark **schrödinger** with **HappyCatOwner**. Inspecting the axioms, we find that (8) is not satisfied by the current representation. Thus, we repair it as required by (8), “inventing” a new element, say , and adding it to D . Accordingly, we stipulate that **schrödinger** is connected to  by an **owns** relation and marking  with **Cat**. We find that, as a consequence of these changes, (8) is satisfied (for the moment). However, the changes have invalidated axioms (5) and (7). We account for the former by introducing a **caresFor** connection from **schrödinger** to . The latter essentially leaves us with two options: we need to mark  either by **Dead** or by **Alive**. This means, we have to make a case distinction and investigate each option separately.

- Let us try and pick **Dead**. Again, examining the axioms, we find (8) violated due to the second part of its consequence. Repairing this requires to mark \mathfrak{a} with **Healthy** which in turn invalidates (6). Hence we have to mark \mathfrak{a} by \neg **Dead**. Unfortunately, we now observe that \mathfrak{a} is marked both by **Dead** and \neg **Dead**, thus we have reached a “dead end” and need to backtrack.
- So, we mark \mathfrak{a} by **Alive**. Also here, we find (8) violated and repair it by marking \mathfrak{a} with **Healthy**, obtaining invalidation of (6) and coping with it by marking \mathfrak{a} by \neg **Dead**. We have thus arrived at a state where our intermediate model satisfies all axioms. Hence, we have obtained a proper model of \mathcal{KB} and conclude that the knowledge base is satisfiable.

However, note that the continued “repairing” performed in a tableau procedure does not necessarily terminate, since performing one repair might cause the need for another repair and so forth ad infinitum.

Example 53. Consider the knowledge base containing the single axiom $\top \sqsubseteq \exists \text{succ}.\top$, which forces every domain element to have a successor. Applying the naïve repair approach from above we will need to introduce a successor for every individual, then successors of successors etc.

Therefore, in order to be applicable as a decision procedure, these infinite computations must be prevented to ensure termination. This can be achieved by a strategy called *blocking*, where certain domain elements are “blocked” (which essentially means that they are exempt from the necessity of being repaired) by other domain individuals which “look the same” in terms of concept memberships. For more advanced DLs, more complicated blocking strategies are needed.

A tableau algorithm for *SHOIQ* is described by Horrocks and Sattler [2007]. A refinement of the tableau technique, called *hypertableau* is at the core of the OWL 2 DL reasoner *HermiT* [Motik *et al.*, 2009c].

8.2 Automata

As mentioned earlier, most DLs satisfy some sort of tree-model property. On the other hand, families of trees (in other words: tree languages) can be represented by appropriate tree-automata. Thus, given an automaton that characterizes the tree models of a knowledge base, the problem of knowledge base satisfiability can be rephrased into the question whether the tree language represented by this corresponding automaton is non-empty. This line of research has been followed by several investigations targeted at standard reasoning as well as conjunctive query answering. Approaches along those lines are e.g. described by Glimm *et al.* [2008a] and Calvanese *et al.* [2009].

Exercise 34. To get a feeling for the relatedness between automata and DL reasoning, try to design an *ALC* knowledge base \mathcal{KB} with the property that for any $r_1, r_2, \dots, r_n \in N_R$ we have that $\mathcal{KB} \models A \sqsubseteq \exists r_1 \exists r_2 \dots \exists r_n. B$ exactly if the word $r_1 r_2 \dots r_n$ matches the regular expression $s^*(rs|srr)^*$.

8.3 Consequence-Based Reasoning

As suggested by their name, consequence-based (also: consequence-driven) reasoning approaches start from the given knowledge base and derive logical consequences of it by means of applying *deduction rules*. A deduction rule has the shape

$$\text{name} \frac{\alpha_1 \cdots \alpha_n}{\alpha}$$

with $\alpha, \alpha_1, \dots, \alpha_n$ being axioms of the underlying logic. To apply a deduction rule means to add α to the set of statements known to be true if truth is already established for $\alpha_1, \dots, \alpha_n$ (be it due to their presence in the knowledge base or because they have been derived by an earlier application of a deduction rule). If, given a set \mathbb{D} of deduction rules, an axiom β can be generated like this from an axiom set $\{\beta_1, \dots, \beta_k\}$ by (possibly manifold) applications of deduction rules, we say that β is *derivable* from $\{\beta_1, \dots, \beta_k\}$ and write $\{\beta_1, \dots, \beta_k\} \vdash \beta$.

In order to be of proper use for the reasoning, the used set \mathbb{D} of deduction rules (also jointly called a *deduction calculus*) has to mimic the logical entailment as defined by the formal semantics. That means that on one hand, β must be a logical consequence of $\{\beta_1, \dots, \beta_k\}$ whenever β is derivable therefrom (in short: $\{\beta_1, \dots, \beta_k\} \vdash \beta$ implies $\{\beta_1, \dots, \beta_k\} \models \beta$) – a property called *soundness* of the deduction calculus. On the other hand, we require its *completeness*, i.e. that every logical consequence of $\{\beta_1, \dots, \beta_k\}$ can also be derived from it (in short: $\{\beta_1, \dots, \beta_k\} \models \beta$ implies $\{\beta_1, \dots, \beta_k\} \vdash \beta$). Sometimes, completeness is constrained to specific axiom types β , e.g. a deduction calculus is called *refutationally complete*, if inconsistency of a knowledge base implies derivability of $\top \sqsubseteq \perp$.

Example 54. The following deduction calculus is sound and refutationally complete for \mathcal{ALC} TBoxes in an appropriate normal form (for details see Simancik *et al.* [2011]). Thereby **A** and **B** denote concept names, H and K are conjunctions of negated and unnegated concept names, whereas M , N , and N_i are disjunctions of concept names.

$$\mathbf{R}_A^+ \frac{}{A \sqcap H \sqsubseteq A}$$

$$\mathbf{R}_A^- \frac{\neg A \sqcap H \sqsubseteq N \sqcup A}{\neg A \sqcap H \sqsubseteq N}$$

$$\mathbf{R}_\sqcap^n \frac{H \sqsubseteq N_1 \sqcup A_1 \cdots H \sqsubseteq N_n \sqcup A_n \quad \prod_{i=1}^n A_i \sqsubseteq M}{H \sqsubseteq M \sqcup \prod_{i=1}^n N_i}$$

$$\mathbf{R}_\exists^+ \frac{H \sqsubseteq N \sqcup A \quad A \sqsubseteq \exists r.B}{H \sqsubseteq N \sqcup \exists r.B}$$

$$\mathbf{R}_\exists^- \frac{H \sqsubseteq N \sqcup \exists r.K \quad K \sqsubseteq N \sqcup A \quad \exists r.A \sqsubseteq B}{H \sqsubseteq M \sqcup B \sqcup \exists r.(K \sqcap \neg A)}$$

$$\mathbf{R}_\perp \frac{H \sqsubseteq N \sqcup \exists r.K \quad K \sqsubseteq \perp}{H \sqsubseteq M}$$

$$\mathbf{R}_\forall \frac{H \sqsubseteq N \sqcup \exists r.K \quad H \sqsubseteq N \sqcup A \quad A \sqsubseteq \forall r.B}{H \sqsubseteq M \sqcup N \sqcup \exists r.(K \sqcap B)}$$

Exercise 35. *Using the above deduction calculus, show that the axiom $D \sqsubseteq G$ can be derived from the knowledge base containing the axioms*

$$A \sqsubseteq B \sqcup C \quad D \sqsubseteq \forall r.A \quad \exists r.B \sqsubseteq E \quad D \sqsubseteq F \sqcup \exists r.\neg C \quad E \sqcap F \sqsubseteq G.$$

However, just a sound and complete deduction calculus is not sufficient for a decision procedure (note that FOL itself has such a calculus while being undecidable). In addition to that, one has to ensure that the “enrichment process” of adding more and more derived consequences to the set of true statements will terminate at some point. One way to guarantee this is to make sure that only finitely many (syntactically) different axioms can be derived. Consequence-driven approaches are described e.g. by Kazakov [2009] and Simancik *et al.* [2011].

8.4 Resolution

Resolution is a technique prominently used in first-order logic theorem proving. At the core of reasoning via resolution is the resolution rule which looks as follows:

$$\text{Res} \frac{A_1 \vee \dots \vee A_i \vee \dots \vee A_n \quad B_1 \vee \dots \vee B_j \vee \dots \vee B_m}{A_1 \vee \dots \vee A_{i-1} \vee A_{i+1} \vee \dots \vee A_n \vee B_1 \vee \dots \vee B_{j-1} \vee B_{j+1} \vee \dots \vee B_m}$$

Thereby, A_k and B_k denote literals, i.e. negated or unnegated FOL atoms and the two literals A_i and B_j are assumed to be complements of each other (i.e. $A_i = \neg B_j$ or $B_j = \neg A_i$). As the resolution rule is a deduction rule, resolution can be seen as a variant of consequence-based reasoning. One of the differences is that resolution is performed not on DL knowledge bases directly but on a FOL translation thereof. Resolution-based methods have been described for DLs up to *SHOIQ* [Motik and Sattler, 2006; Kazakov and Motik, 2008].

9 Description Logics and OWL

*In fact, in terms of syntax, OWL
Just tends to be a bulky fowl,
However, if it mates with Turtle
This union turns out rather fertile;
I deem the offspring of this love
As graceful as a turtledove.*

As mentioned before, the web ontology language OWL is based on Description Logics but also features additional types of extra-logical information, concerning, e.g., ontology versioning information and annotations. Moreover, OWL supports modeling and reasoning with datatypes which we omitted from our consideration. Likewise, keys are supported in OWL but not discussed here.

In this section, we will see how any OWL DL compliant reasoning tool can be used to decide *SROIQ* knowledge base satisfiability as well as any other reasoning task which can be reduced to it.

“OWL speak” differs partially from the terms normally used in description logics. The following table gives a synopsis of the corresponding terms used in the OWL vs. the DL community as well as in the domain of classical first-order logic.

OWL	DL	FOL
class name	concept name	unary predicate
class	concept	formula with one free variable
object property name	role name	binary predicate
object property	role	formula with two free variables
ontology	knowledge base	theory
axiom	axiom	sentence
vocabulary	vocabulary / signature	signature

In the next two sections, we briefly explain how a *SRQIQ* knowledge base can be translated into an OWL 2 DL ontology such that satisfiability and entailment checks can be performed by OWL reasoning engines.

9.1 Translating DL KBs into OWL

For translating a *SRQIQ* knowledge base into an OWL ontology, some technical issues need to be taken care of. First of all, both the used vocabulary as well as the constructors have to be URIs (i.e. uniform resource identifiers, that is, terms following the prescribed naming scheme prevalent in the Semantic Web). The URIs for the used individual, concept, and role names can be chosen rather arbitrarily, while the URIs for constructors etc. are prescribed and associated to specific namespaces usually associated to the prefixes `owl:`, `rdfs:`, `rdf:`, and `xsd:`. For the sake of simplicity, we will assume that all used individual, concept and role names from the DL knowledge base are syntactically well-formed URIs.

Second, the mainly used encoding of OWL is as an RDF document [Manola and Miller, 2004]. On one hand, this is advantageous from a downward compatibility and tool interoperability point of view; in fact the encoding of concept and role assertions in OWL and RDF coincide and some other RDFS statements are available in OWL as well with a similar semantics. On the other hand, the encoding as RDF also imposes some restrictions on the way logical axioms can be encoded. As RDF is a graph-based formalism consisting of node-edge-node triples, DL axioms and complex concepts have to be transformed into a graph-like representation. This is done by virtue of the typical means used to encode complex structures in RDF: structural bnodes and graph-based encoding of lists.

For our treatise, we will use the Turtle [Beckett and Berners-Lee, 14 January 2008] notation, which seems most appropriate as it illustrates the underlying RDF triple

structure while at the same time hiding the very low-level details (such as the triplification of the list structures employed for the RDF encoding of OWL).

The translation of a *SRIOQ* knowledge base \mathcal{KB} contains three parts: a preamble containing the definition of namespaces, declarations of the used concept (resp. class) and role (resp. object property) names, and finally a part containing the OWL counterparts of the axioms from \mathcal{KB} . Hence, we let

$$[\mathcal{KB}] = \text{Pre} + \text{Dec}(\mathcal{KB}) + \sum_{\alpha \in \mathcal{KB}} [\alpha]$$

where $+$ denotes concatenation of strings. Thereby the preamble is defined by

$$\text{Pre} = \begin{cases} \text{@prefix owl:} <\text{http://www.w3.org/2002/07/owl\#}> . \\ \text{@prefix rdfs:} <\text{http://www.w3.org/2000/01/rdf-schema\#}> . \\ \text{@prefix rdf:} <\text{http://www.w3.org/1999/02/22-rdf-syntax-ns\#}> . \\ \text{@prefix xsd:} <\text{http://www.w3.org/2001/XMLSchema\#}> . \end{cases}$$

whereas the declarations are expressed by according typing statements:

$$\begin{aligned} \text{Dec}(\mathcal{KB}) = & \sum_{A \in \mathcal{N}_C(\mathcal{KB})} A \text{ rdf:type owl:Class} . \\ & + \sum_{r \in \mathcal{N}_R(\mathcal{KB})} r \text{ rdf:type owl:ObjectProperty} . \end{aligned}$$

As displayed above, the actual knowledge base is translated axiom-wise via the function $[\cdot]$ defined on the next page. The latter makes calls to the functions $[\cdot]_{\mathbf{C}}$ and $[\cdot]_{\mathbf{R}}$ given further below, which are used to decompose and recursively translate complex concepts and roles, respectively.

$$\begin{aligned} [r_1 \circ \dots \circ r_n \sqsubseteq r] &= [r]_{\mathbf{R}} \text{ owl:propertyChainAxiom } ([r_1]_{\mathbf{R}} \dots [r_n]_{\mathbf{R}}) . \\ [\text{Dis}(r, r')] &= [r]_{\mathbf{R}} \text{ owl:propertyDisjointWith } [r']_{\mathbf{R}} . \\ [C \sqsubseteq D] &= [C]_{\mathbf{C}} \text{ rdfs:subClassOf } [D]_{\mathbf{C}} . \\ [C(a)] &= a \text{ rdf:type } [C]_{\mathbf{C}} . \\ [r(a, b)] &= a \text{ } r \text{ } b . \\ [r^-(a, b)] &= b \text{ } r \text{ } a . \\ [\neg r(a, b)] &= [] \text{ rdf:type owl:NegativePropertyAssertion ;} \\ &\quad \text{owl:assertionProperty } [r]_{\mathbf{R}} ; \\ &\quad \text{owl:sourceIndividual } a ; \text{ owl:targetValue } b . \\ [a \approx b] &= a \text{ owl:sameAs } b . \\ [a \not\approx b] &= a \text{ owl:differentFrom } b . \end{aligned}$$

$$\begin{aligned}
 [u]_{\mathbf{R}} &= \text{owl:topObjectProperty} \\
 [r]_{\mathbf{R}} &= r \\
 [r -]_{\mathbf{R}} &= [\text{owl:inverseOf } :r] \\
 [A]_{\mathbf{C}} &= A \\
 [\top]_{\mathbf{C}} &= \text{owl:Thing} \\
 [\perp]_{\mathbf{C}} &= \text{owl:Nothing} \\
 [\{a_1, \dots, a_n\}]_{\mathbf{C}} &= [\text{rdf:type owl:Class ; owl:oneOf (:a}_1 \dots :a_n)] \\
 [\neg C]_{\mathbf{C}} &= [\text{rdf:type owl:Class ; owl:complementOf } [C]_{\mathbf{C}}] \\
 [C_1 \cap \dots \cap C_n]_{\mathbf{C}} &= [\text{rdf:type owl:Class ; owl:intersectionOf (} [C_1]_{\mathbf{C}} \dots [C_n]_{\mathbf{C}})] \\
 [C_1 \sqcup \dots \sqcup C_n]_{\mathbf{C}} &= [\text{rdf:type owl:Class ; owl:unionOf (} [C_1]_{\mathbf{C}} \dots [C_n]_{\mathbf{C}})] \\
 [\exists r.C]_{\mathbf{C}} &= [\text{rdf:type owl:Restriction ;} \\
 &\quad \text{owl:onProperty } [r]_{\mathbf{R}} \text{ ; owl:someValuesFrom } [C]_{\mathbf{C}}] \\
 [\forall r.C]_{\mathbf{C}} &= [\text{rdf:type owl:Restriction ;} \\
 &\quad \text{owl:onProperty } [r]_{\mathbf{R}} \text{ ; owl:allValuesFrom } [C]_{\mathbf{C}}] \\
 [\exists r.\text{Self}]_{\mathbf{C}} &= [\text{rdf:type owl:Restriction ;} \\
 &\quad \text{owl:onProperty } [r]_{\mathbf{R}} \text{ ; owl:hasSelf "true"^^xsd:boolean }] \\
 [\geq n r.C]_{\mathbf{C}} &= [\text{rdf:type owl:Restriction ;} \\
 &\quad \text{owl:minQualifiedCardinality } n \text{^^xsd:nonNegativeInteger ;} \\
 &\quad \text{owl:onProperty } [r]_{\mathbf{R}} \text{ ; owl:onClass } [C]_{\mathbf{C}}] \\
 [\leq n r.C]_{\mathbf{C}} &= [\text{rdf:type owl:Restriction ;} \\
 &\quad \text{owl:maxQualifiedCardinality } n \text{^^xsd:nonNegativeInteger ;} \\
 &\quad \text{owl:onProperty } [r]_{\mathbf{R}} \text{ ; owl:onClass } [C]_{\mathbf{C}}]
 \end{aligned}$$

Example 55. For the knowledge base \mathcal{KB} from Example 12, the translation $[\mathcal{KB}]$ looks as follows (for better readability, we use the namespace `http://www.example.org/#` for individual, concept, and role names and abbreviate it by the empty prefix as shown in the first line of the translation):

```

@prefix :      <http://www.example.org/#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .

:owns          rdf:type owl:ObjectProperty .
:caresFor      rdf:type owl:ObjectProperty .
:Cat           rdf:type owl:Class .
:Dead          rdf:type owl:Class .
:Alive         rdf:type owl:Class .
:Healthy       rdf:type owl:Class .
:HappyCatOwner rdf:type owl:Class .
    
```

```

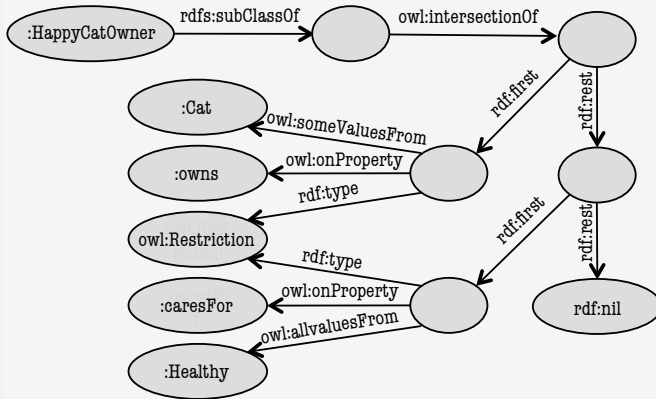
:owns      rdfs:subPropertyOf :caresFor .

:Healthy   rdfs:subClassOf [ owl:complementOf :Dead ] .
:Cat       rdfs:subClassOf [ owl:unionOf (:Dead :Alive) ] .
:HappyCatOwner rdfs:subClassOf
  [ owl:intersectionOf
    ( [ rdf:type owl:Restriction ;
      owl:onProperty :owns ; owl:someValuesFrom :Cat ]
      [ rdf:type owl:Restriction ;
      owl:onProperty :caresFor ; owl:allValuesFrom :Healthy ] )
  ] .

:schrödinger rdf:type :HappyCatOwner .

```

To give an idea, how the RDF graph representation of an OWL ontology looks like, the last TBox axiom is displayed graphically in the following picture.



Exercise 36. Translate the knowledge base from Example 21 and the initial axiom from Example 33 into OWL ontologies in Turtle syntax.

9.2 Expressing OWL Axioms in *SRDIQ*

In fact, the OWL specification features much more axiom types than the ones used above to translate *SRDIQ* knowledge bases. As far as the purely logical axioms are concerned (i.e. excluding everything referring to datatypes, keys, annotations, or the like), all these axioms can be considered as *syntactic sugar*, i.e., they can be conceived as shortcuts for other axioms expressed in the “core” OWL language used in the definitions above. In the sequel, we give the DL paraphrases of these axioms

Axiom type	Turtle notation	DL paraphrase
Class Equivalence	$\llbracket C \rrbracket_{\mathcal{C}}$ owl:equivalentClass $\llbracket D \rrbracket_{\mathcal{C}}$.	$C \sqsubseteq D, D \sqsubseteq C$
Class Disjointness	$\llbracket C \rrbracket_{\mathcal{C}}$ owl:disjointWith $\llbracket D \rrbracket_{\mathcal{C}}$.	$C \sqcap D \sqsubseteq \perp$
Disjoint Classes	\square rdf:type owl:AllDisjointClasses ; owl:members ($\llbracket C_1 \rrbracket_{\mathcal{C}}$... $\llbracket C_n \rrbracket_{\mathcal{C}}$) .	$C_i \sqcap C_j \sqsubseteq \perp$ for all $1 \leq i < j \leq n$
Disjoint Union	$\llbracket C \rrbracket_{\mathcal{C}}$ owl:disjointUnionOf ($\llbracket C_1 \rrbracket_{\mathcal{C}}$... $\llbracket C_n \rrbracket_{\mathcal{C}}$) .	$\bigsqcup_{i < j} C_i \sqsubseteq C$ $C_i \sqcap C_j \sqsubseteq \perp$ for all $1 \leq i < j \leq n$
Property Equivalence	$\llbracket r \rrbracket_{\mathcal{R}}$ owl:equivalentProperty $\llbracket s \rrbracket_{\mathcal{R}}$.	$r \sqsubseteq s, s \sqsubseteq r$
Disjoint Properties	\square rdf:type owl:AllDisjointProperties ; owl:members ($\llbracket r_1 \rrbracket_{\mathcal{R}}$... $\llbracket r_n \rrbracket_{\mathcal{R}}$) .	$\text{Dis}(r_i, r_j)$ for all $1 \leq i < j \leq n$
Inverse Properties	$\llbracket r \rrbracket_{\mathcal{R}}$ owl:inverseOf $\llbracket s \rrbracket_{\mathcal{R}}$.	$\text{Inv}(r) \sqsubseteq s$
Property Domain	$\llbracket r \rrbracket_{\mathcal{R}}$ rdfs:domain $\llbracket C \rrbracket_{\mathcal{C}}$.	$\exists r. \top \sqsubseteq C$
Property Range	$\llbracket r \rrbracket_{\mathcal{R}}$ rdfs:range $\llbracket C \rrbracket_{\mathcal{C}}$.	$\top \sqsubseteq \forall r. C$
Functional Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:FunctionalProperty .	$\top \sqsubseteq \leq 1r. \top$
Inverse Functional Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:InverseFunctionalProperty .	$\top \sqsubseteq \leq 1\text{Inv}(r). \top$
Reflexive Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:ReflexiveProperty .	$\top \sqsubseteq \exists r. \text{Self}$
Irreflexive Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:IrreflexiveProperty .	$\exists r. \text{Self} \sqsubseteq \perp$
Symmetric Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:SymmetricProperty .	$\text{Inv}(r) \sqsubseteq r$
Asymmetric Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:AsymmetricProperty .	$\text{Dis}(\text{Inv}(r), r)$
Transitive Property	$\llbracket r \rrbracket_{\mathcal{R}}$ rdf:type owl:TransitiveProperty .	$r \circ r \sqsubseteq r$
Different Individuals	\square rdf:type owl:AllDifferent ; owl:members (a_1 ... a_n) .	$a_i \not\approx a_j$ for all $1 \leq i < j \leq n$

10 Further Reading

At the end of this chapter, we give a few pointers to further reading with respect to different aspects of the contents presented here. Note that this list is certainly incomplete and subject to personal inclinations.

Description Logics. As central reference to the area of Description Logics, the primary resource is certainly the Description Logic Handbook [Baader *et al.*, 2007], providing an overview of the subject, introductory parts as well as parts dedicated to advanced issues. The description logic *SR_OI_Q* which, together with its sublogics, was the main subject of our treatise is introduced by Horrocks *et al.* [2006], the according reasoning complexity results are established by Kazakov [2008].

Conjunctive Queries in Description Logics. While the principled problem of reasoning in DLs up to *SR_OI_Q* can be considered to be solved, conjunctive query answering is still a subject of active research and only preliminary decidability and complexity results are available. Most notably, decidability of *SR_OI_Q* and even of *SH_OI_Q* is unsolved. On the other hand, the problem is settled

for *SHIQ* [Glimm *et al.*, 2008c] and *SHOQ* [Glimm *et al.*, 2008b]. Moreover, Calvanese *et al.* [2009] captured additionally *SHOI* and extended the results to regular path queries. The most expressive Boolean-closed DL simultaneously featuring nominal concepts, inverses and number restrictions (i.e., \mathcal{O} , \mathcal{I} , and \mathcal{Q}) for which decidability is known is *ALCHOIQb* [Rudolph and Glimm, 2010].

Relations to Logics in General. For foundations of logics, the textbook by Schöning [2008] is certainly a good starting point in particular for computer scientists, whereas Ebbinghaus *et al.* [1996] capture mathematical aspects. Model theory is treated in depth by Chang and Keisler [1990]. For an introduction into the area of theorem proving in a first-order logic setting, we recommend the textbook by Fitting [1996]. We suggest to consult Papadimitriou [1994] for the study of algorithmic complexity theory.

The correspondence of DLs and first-order logic (in particular the 2-variable fragment) has e.g. been described by Borgida [1996], the complexity treatment on the 2-variable fragment of FOL with counting quantifiers by Pratt-Hartmann [2005] has served as basis for a row of DL complexity results. The relatedness of DLs with modal logics (see the textbook by Blackburn *et al.* [2006] for a thorough introduction) is treated by Schild [1991]. As another closely related logic, the guarded fragment of FOL is described by Andr eka *et al.* [1998].

AI and Knowledge Representation. A central reference for a comprehensive overview of the area of AI as a whole is the seminal textbook by Russell and Norvig [2003]. Knowledge Representation in particular is treated by Sowa [1984] and van Harmelen *et al.* [2008].

Semantic Web and OWL. The Semantic Web vision is described in the seminal paper by Berners-Lee *et al.* [2001]. In order to get an overview over all aspects of (Web) ontologies, the Ontology Handbook by Staab and Studer [2009] is a central reference.

As far as technical questions about syntax and semantics of OWL is concerned, the primary resource are the W3C Recommendation Documents. Next to an overview [OWL Working Group, 2009], syntax and semantics are treated by Motik *et al.* [2009b] and Motik *et al.* [2009a], respectively, whereas Patel-Schneider and Motik [2009] tackle the RDF serialization of OWL. The OWL 2 Primer by Hitzler *et al.* [2009a] gives an informal introduction into the use of OWL. A thorough treatment of all the standardized Semantic Web formalisms is provided by the textbook Foundations of Semantic Web Technologies [Hitzler *et al.*, 2009b].

Acknowledgements. I thank all people who helped me in one or the other way to accumulate the knowledge about DLs which I gave a partial overview of in this lecture. I am grateful to the organizers of the Reasoning Web Summer School 2011 for giving me the opportunity to teach. I thank the anonymous reviewers for their comments on an earlier version of this document. I am indebted to Anees ul Mehdi, Nadeschda Nikitina and Jens Wissmann for their thorough proofreading.

Special thanks go to Ian Horrocks for his valuable feedback in terms of poetic quality assurance. The DL logo displayed at the beginning of the chapter goes back to Enrico Franconi, the deduction calculus for \mathcal{ALC} comes from Yevgeny Kazakov. Further inspiration was drawn from (in alphabetical order) Benedict XVI, Nicolas Chamfort, Edward Lear, the Rolling Stones, William Shakespeare and W.A. Spooner.

References

- [Andréka *et al.*, 1998] Andréka, H., van Benthem, J.F.A.K., Németi, I.: Modal languages and bounded fragments of predicate logic. *Journal of Philosophical Logic* 27(3), 217–274 (1998)
- [Baader *et al.*, 2007] Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): *The Description Logic Handbook: Theory, Implementation, and Applications*, 2nd edn. Cambridge University Press, Cambridge (2007)
- [Beckett and Berners-Lee, 14 January 2008] Beckett, D., Berners-Lee, T.: Turtle – Terse RDF Triple Language. W3C Team Submission (January 14, 2008), <http://www.w3.org/TeamSubmission/turtle/>
- [Berners-Lee *et al.*, 2001] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. In: *Scientific American*, pp. 96–101 (May 2001)
- [Blackburn *et al.*, 2006] Blackburn, P., van Benthem, J.F.A.K., Wolter, F. (eds.): *Handbook of Modal Logic. Studies in Logic and Practical Reasoning*, vol. 3. Elsevier Science, Amsterdam (2006)
- [Borgida, 1996] Borgida, A.: On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence* 82(1–2), 353–367 (1996)
- [Brachman and Levesque, 1984] Brachman, R.J., Levesque, H.J.: The tractability of subsumption in frame-based description languages. In: Brachman, R.J. (ed.) *Proceedings of the 4th National Conference on Artificial Intelligence (AAAI 1984)*, pp. 34–37. AAAI Press, Menlo Park (1984)
- [Calvanese *et al.*, 2009] Calvanese, D., Eiter, T., Ortiz, M.: Regular path queries in expressive description logics with nominals. In: Boutilier, C. (ed.) *Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI 2009)*, pp. 714–720 (2009)
- [Chandra and Merlin, 1977] Chandra, A.K., Merlin, P.M.: Optimal implementation of conjunctive queries in relational data bases. In: Hopcroft, J.E., Friedman, E.P., Harrison, M.A. (eds.) *Proceedings of the 9th Annual ACM Symposium on Theory of Computing (STOC 1977)*, pp. 77–90. ACM Press, New York (1977)
- [Chang and Keisler, 1990] Chang, C.C., Jerome Keisler, H.: *Model Theory*, 3rd edn. *Studies in Logic and the Foundations of Mathematics*, vol. 73. North Holland, Amsterdam (1990)
- [Ebbinghaus *et al.*, 1996] Ebbinghaus, H.-D., Flum, J., Thomas, W.: *Mathematical Logic*. Springer, Heidelberg (1996)
- [Fitting, 1996] Fitting, M.: *First-Order Logic and Automated Theorem Proving*, 2nd edn. Springer, Heidelberg (1996)
- [Ganter and Wille, 1997] Ganter, B., Wille, R.: *Formal Concept Analysis: Mathematical Foundations*. Springer, Heidelberg (1997)
- [Glimm *et al.*, 2008a] Glimm, B., Horrocks, I., Sattler, U.: Deciding \mathcal{SHOQ}^{\cap} knowledge base consistency using alternating automata. In: Baader, F., Lutz, C., Motik, B. (eds.) *Description Logics. CEUR Workshop Proceedings*, vol. 353 (2008), CEUR-WS.org

- [Glimm *et al.*, 2008b] Glimm, B., Horrocks, I., Sattler, U.: Unions of conjunctive queries in *SHOQ*. In: Brewka, G., Lang, J. (eds.) Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 252–262. AAAI Press, Menlo Park (2008)
- [Glimm *et al.*, 2008c] Glimm, B., Lutz, C., Horrocks, I., Sattler, U.: Answering conjunctive queries in the SHIQ description logic. *Journal of Artificial Intelligence Research* 31, 150–197 (2008)
- [Golbreich *et al.*, 2006] Golbreich, C., Zhang, S., Bodenreider, O.: The foundational model of anatomy in OWL: Experience and perspectives. *Journal of Web Semantics* 4(3) (2006)
- [Haarslev and Möller, 2001] Haarslev, V., Möller, R.: RACER System Description. In: Goré, R.P., Leitsch, A., Nipkow, T. (eds.) IJCAR 2001. LNCS (LNAI), vol. 2083, pp. 701–705. Springer, Heidelberg (2001)
- [Hitzler *et al.*, 2009a] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S. (eds.): OWL 2 Web Ontology Language: Primer. W3C Recommendation (2009), <http://www.w3.org/TR/owl2-primer/>
- [Hitzler *et al.*, 2009b] Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)
- [Horridge *et al.*, 2008] Horridge, M., Parsia, B., Sattler, U.: Laconic and Precise Justifications in OWL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 323–338. Springer, Heidelberg (2008)
- [Horrocks and Sattler, 2007] Horrocks, I., Sattler, U.: A tableau decision procedure for *SHOIQ*. *Journal of Automated Reasoning* 39(3), 249–276 (2007)
- [Horrocks *et al.*, 2006] Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible *SROIQ*. In: Doherty, P., Mylopoulos, J., Welty, C.A. (eds.) Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), pp. 57–67. AAAI Press, Menlo Park (2006)
- [Kazakov and Motik, 2008] Kazakov, Y., Motik, B.: A resolution-based decision procedure for *SHOIQ*. *Journal of Automated Reasoning* 40(2-3), 89–116 (2008)
- [Kazakov, 2008] Kazakov, Y.: *RIQ* and *SROIQ* are harder than *SHOIQ*. In: Brewka, G., Lang, J. (eds.) Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR 2008), pp. 274–284. AAAI Press, Menlo Park (2008)
- [Kazakov, 2009] Kazakov, Y.: Consequence-driven reasoning for horn *SHIQ* ontologies. In: Boutilier, C. (ed.) Proceedings of the 21st International Conference on Artificial Intelligence (IJCAI 2009), pp. 2040–2045 (2009)
- [Krötzsch *et al.*, 2008] Krötzsch, M., Rudolph, S., Hitzler, P.: Description logic rules. In: Ghallab, M., Spyropoulos, C.D., Fakotakis, N., Avouris, N. (eds.) Proceedings of the 18th European Conference on Artificial Intelligence (ECAI 2008), pp. 80–84. IOS Press, Amsterdam (2008)
- [Lehmann, 2009] Lehmann, J.: DL-learner: Learning concepts in description logics. *Journal of Machine Learning Research* 10, 2639–2642 (2009)
- [Lloyd and Topor, 1984] Lloyd, J.W., Topor, R.W.: Making prolog more expressive. *Journal of Logic Programming* 1(3), 225–240 (1984)
- [Lutz, 2008] Lutz, C.: The complexity of conjunctive query answering in expressive description logics. In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 179–193. Springer, Heidelberg (2008)
- [Manola and Miller, 2004] Manola, F., Miller, E. (eds.): Resource Description Framework (RDF): Primer. W3C Recommendation (2004), <http://www.w3.org/TR/rdf-primer/>

- [Minsky, 1974] Minsky, M.: A framework for representing knowledge. Artificial intelligence memo, A.I. Laboratory. Massachusetts Institute of Technology, Cambridge (1974)
- [Motik and Sattler, 2006] Motik, B., Sattler, U.: A Comparison of Reasoning Techniques for Querying Large Description Logic ABoxes. In: Hermann, M., Voronkov, A. (eds.) LPAR 2006. LNCS (LNAI), vol. 4246, pp. 227–241. Springer, Heidelberg (2006)
- [Motik *et al.*, 2009a] Motik, B., Patel-Schneider, P.F., Grau, B.C. (eds.): OWL 2 Web Ontology Language: Direct Semantics. W3C Recommendation (2009), <http://www.w3.org/TR/owl2-direct-semantics/>
- [Motik *et al.*, 2009b] Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (2009), <http://www.w3.org/TR/owl2-syntax/>
- [Motik *et al.*, 2009c] Motik, B., Shearer, R., Horrocks, I.: Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research (JAIR)* 36, 165–228 (2009)
- [Noia *et al.*, 2009] Di Noia, T., Di Sciascio, E., Donini, F.M.: A tableaux-based calculus for abduction in expressive description logics: Preliminary results. In: Grau, B.C., Horrocks, I., Motik, B., Sattler, U. (eds.) *Description Logics*. CEUR Workshop Proceedings, vol. 477 (2009), [CEUR-WS.org](http://www.w3.org)
- [OWL Working Group, 2009] W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (2009) <http://www.w3.org/TR/owl2-overview/>
- [Papadimitriou, 1994] Papadimitriou, C.H.: *Computational Complexity*. Addison-Wesley, Reading (1994)
- [Patel-Schneider and Motik, 2009] Patel-Schneider, P.F., Motik, B. (eds.): OWL 2 Web Ontology Language: Mapping to RDF Graphs. W3C Recommendation (2009), <http://www.w3.org/TR/owl2-mapping-to-rdf/>
- [Pratt-Hartmann, 2005] Pratt-Hartmann, I.: Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Language and Information* 14, 369–395 (2005)
- [Quillian, 1968] Ross Quillian, M.: Semantic memory. In: Minsky, M. (ed.) *Semantic Information Processing*, ch. 4, pp. 227–270. MIT Press, Cambridge (1968)
- [Rudolph and Glimm, 2010] Rudolph, S., Glimm, B.: Nominals, inverses, counting, and conjunctive queries or: Why infinity is your friend! *Journal of Artificial Intelligence Research (JAIR)* 39, 429–481 (2010)
- [Rudolph *et al.*, 2008] Rudolph, S., Krötzsch, M., Hitzler, P.: All elephants are bigger than all mice. In: Baader, F., Lutz, C., Motik, B. (eds.) *Proceedings of the 21st International Workshop on Description Logics (DL 2008)*. CEUR Workshop Proceedings, vol. 353 (2008), [CEUR-WS.org](http://www.w3.org)
- [Rudolph *et al.*, 2008b] Rudolph, S., Krötzsch, M., Hitzler, P.: Terminological reasoning in SHIQ with ordered binary decision diagrams. In: *Proceedings of the 23rd AAAI Conference on Artificial Intelligence (AAAI 2008)*, pp. 529–534. AAAI Press, Menlo Park (2008)
- [Rudolph, 2004] Rudolph, S.: Exploring relational structures via FLE. In: Wolff, K.E., Pfeiffer, H.D., Delugach, H.S. (eds.) *ICCS 2004*. LNCS (LNAI), vol. 3127, pp. 196–212. Springer, Heidelberg (2004)
- [Russell and Norvig, 2003] Russell, S., Norvig, P.: *Artificial Intelligence: A Modern Approach*, 2nd edn. Prentice Hall, Englewood Cliffs (2003)

- [Schild, 1991] Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Mylopoulos, J., Reiter, R. (eds.) Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI 1991), pp. 466–471. Morgan Kaufmann, San Francisco (1991)
- [Schmidt-Schauß and Smolka, 1991] Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. *Journal of Artificial Intelligence* 48, 1–26 (1991)
- [Schöning, 2008] Schöning, U.: *Logic for Computer Scientists*. Birkhäuser, Basel (2008)
- [Shearer and Horrocks, 2009] Shearer, R., Horrocks, I.: Exploiting Partial Information in Taxonomy Construction. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 569–584. Springer, Heidelberg (2009)
- [Sidhu *et al.*, 2005] Sidhu, A., Dillon, T., Chang, E., Sidhu, B.S.: Protein ontology development using OWL. In: Proceedings of the 1st OWL Experiences and Directions Workshop (OWLED 2005). CEUR Workshop Proceedings, vol. 188 (2005), <http://ceur-ws.org/>
- [Simancik *et al.*, 2011] Simancik, F., Kazakov, Y., Horrocks, I.: Consequence-based reasoning beyond horn ontologies. In: Walsh, T. (ed.) Proceedings of the 22nd International Conference on Artificial Intelligence, IJCAI 2011 (2011)
- [Sirin *et al.*, 2007] Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics* 5(2), 51–53 (2007)
- [Sowa, 1984] Sowa, J.F.: *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley, Reading (1984)
- [Staab and Studer, 2009] Staab, S., Studer, R. (eds.): *Handbook on Ontologies*, 2nd edn. International Handbooks on Information Systems. Springer, Heidelberg (2009)
- [Stuckenschmidt *et al.*, 2009] Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*. LNCS, vol. 5445. Springer, Heidelberg (2009)
- [Tsarkov and Horrocks, 2006] Tsarkov, D., Horrocks, I.: FaCT++ Description Logic Reasoner: System Description. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 292–297. Springer, Heidelberg (2006)
- [van Harmelen *et al.*, 2008] van Harmelen, F., Lifschitz, V., Porter, B.: *Handbook of Knowledge Representation. Foundations of Artificial Intelligence*. Elsevier, Amsterdam (2008)
- [Wolstencroft *et al.*, 2005] Wolstencroft, K., Brass, A., Horrocks, I., Lord, P., Sattler, U., Turi, D., Stevens, R.: A little semantic web goes a long way in biology. In: Gil, Y., Motta, E., Benjamins, V.R., Musen, M.A. (eds.) ISWC 2005. LNCS, vol. 3729, pp. 786–800. Springer, Heidelberg (2005)