Chapter 2
Background

Abstract Description Logics is a family of knowledge representation formalisms used to represent knowledge of a domain, usually called "world". For that, it first defines the relevant concepts of the domain—"terminology"—and then, using these concepts, specifies properties of objects and individuals of that domain. In this chapter, we review the syntax, semantics and the main logical properties of DL that we will use in the following chapters.

Keywords Description logic · ALC · Logic · Syntax · Semantics · Proof theory · Axiomatization · Theory · Satisfiable · Tautology

2.1 A Basic Description Logic

Comparing to its predecessors formalisms, Description Logics are equipped with a formal, logic-based semantics. Description Logics differ each other from the constructors they provide. Concept constructors are used to build more complex descriptions of concepts from *atomic concepts* and role constructor to build complex role descriptions from *atomic roles*.

$\mathcal{ALC}$ is a basic Description Logics [1] and its syntax of concept descriptions is as following:

$$\phi_c \rightarrow \top \mid \bot \mid A \mid \neg\phi_c \mid \phi_c \sqcap \phi_c \mid \phi_c \sqcup \phi_c \mid \exists R.\phi_c \mid \forall R.\phi_c$$

where A stands for atomic concepts and R for atomic roles. The concepts $\bot$ and $\top$ could be omitted since they are just abbreviations for $\alpha \sqcap \neg\alpha$ and $\alpha \sqcup \neg\alpha$ for any given concept description α.

The semantics of concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, .^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ of $.^{\mathcal{I}}$ is a non-empty set of individuals and the interpretation function $.^{\mathcal{I}}$ maps each atomic concept A to a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and for each atomic role a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The function $\cdot^{\mathcal{I}}$ is extended to concept descriptions inductive as follows:

$$\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}
\end{aligned}$$

Knowledge representation systems based on description logics provide various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. One of the most important inference services of DL systems is computing the subsumption hierarchy of a given finite set of concept descriptions.

Definition 1 The concept description D subsumes the concept description C, written $C \sqsubseteq D$, if and only if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all interpretations $\mathcal{I}$.

Definition 2 C is satisfiable if and only if there is an interpretation $\mathcal{I}$ such that $C^{\mathcal{I}} \neq \emptyset$.

Definition 3 C is valid or a tautology if and only if, for all interpretation $\mathcal{I}$, $C^{\mathcal{I}} \equiv \Delta^{\mathcal{I}}$.

Definition 4 C and D are equivalent, written $C \equiv D$, if and only if $C \sqsubseteq D$ and $D \sqsubseteq C$.

We used to call $C \sqsubseteq D$ and $C \equiv D$ *terminological axioms*. Axioms of the first kind are called *inclusions*, while axioms of the second kind are called *equalities*. If an interpretation satisfies an axiom (or a set of axioms), then we say that is a *model* of this axiom (or set of axioms).

An equality axiom whose left-hand side is an atomic concept is a *definition*. Definitions are used to introduce *names* for complex descriptions. For instance, the axiom

$$\mathit{Mother} \equiv \mathit{Woman} \sqcap \exists\ \mathit{hasChild}.\mathit{Person}$$

associates to the description on the right-hand side the name *Mother*.

A finite set of definitions $\mathcal{T}$ where no symbolic name is defined more than once is called a *terminology* or *TBox*. In other words, for every atomic concept A there is at most one axiom in $\mathcal{T}$ whose left-hand side is A. Given a $\mathcal{T}$, we divide the atomic concepts occurring in it into two sets, the *name symbols* $\mathcal{N}_{\mathcal{T}}$ that occur on the left-hand side of some axiom and the *base symbols* $\mathcal{B}_{\mathcal{T}}$ that occur only on the right-hand side of axioms. Name symbols are also called *defined* concepts and base symbols *primitive* concepts. The terminology should *define* the name symbols in terms of the base symbols.

With the definitions of the previous paragraphs, we must also extend the definitions of *interpretations* to deal with TBox. A *base interpretation* $\cdot^{\mathcal{I}}$ for $\mathcal{T}$ is an interpretation just for the base symbols. An interpretation that also interprets the name symbols is called an *extension* of $\cdot^{\mathcal{I}}$. There are much more to say about such extensions. For instance, whenever we have cyclic definitions in a *TBox* the *descriptive* semantics given so far is not sufficient. In that case, we usually work with *fixpoint semantics*, we cite [1] for a complete reference.

2.2 Individuals

Besides the TBox component, in a knowledge base we usually have to describe individuals and assertions about them. We call the set of assertions about individual in a knowledge base a *world description* or *ABox*. In a ABOX we introduce individuals and describe their properties using the roles and concepts introduced or defined in the TBox. We have two kind of formulas to express assertions about individuals:

$$C(a) \qquad R(b, c)$$

The formula on the left is called *concept assertion*. It states that the individual a belongs to the interpretation of the concept C. The formula on the right is called *role assertion* that states that the individual c is a filler of the role R for b. Following the typical example from [1], if *Father* is a concept name and *hasChild* a role name, then we can have the following assertions about individual named *Peter*, *Paul*, *Mary*:

$$Father(Peter) \qquad hasChild(Mary, Paul)$$

The meaning of the left assertion is that *Peter* is a father and the assertion on the right says that *Paul* is a child of *Mary*.

Once more we have to extend the notion of interpretation in order to provide semantics to ABoxes. Essentially, the interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ besides mapping concepts to sets and roles to binary relations, also maps individual names a to an element $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. We usually assume that distinct names denote distinct objects, this is called the *unique name assumption* (UNA). Formally, if a and b are distinct names, then $a^{\mathcal{I}} \neq b^{\mathcal{I}}$.

An interpretation $\mathcal{I}$ satisfy the assertion $C(a)$, if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and the role assertion $R(a, b)$, if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. In that cases, we write:

$$\mathcal{I} \models C(a) \qquad \mathcal{I} \models R(a, b)$$

An interpretation satisfies an ABox if it satisfies each assertions on it, that is, it is a *model* for the ABox. An interpretation that satisfies an ABox with respect to a TBox whenever it is a model for both.

2.3 Description Logics Family

If we add to $\mathcal{ALC}$ more constructors, more expressivity power to describe concepts and roles we obtain. Description logics are a huge family of logics, it is not our goal to present and discuss all of them. We will describe in this section only the extensions of $\mathcal{ALC}$ that we will deal in this book. For a complete reference we indicate [1].[1]

Two of the most useful extensions of $\mathcal{ALC}$ is $\mathcal{ALCN}$ and $\mathcal{ALCQ}$. $\mathcal{ALCN}$ includes *number restrictions* written as $\leq nR$ or $\geq nR$ where n ranges over non-negative integers. $\mathcal{ALCQ}$ allows constructors for qualified number restrictions of the form $\leq nR.C$ and $\geq nR.C$. The semantics of those constructors are given by the definitions below.

$$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \leq n\}$$
$$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}}\}| \geq n\}$$
$$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}} \land b \in C^{\mathcal{I}}\}| \leq n\}$$
$$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b) \in R^{\mathcal{I}} \land b \in C^{\mathcal{I}}\}| \geq n\}$$

We name $\mathcal{AL}$-languages using letters to indicate the allowed constructor:

$$\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{Q}][\mathcal{C}]$$

The $\mathcal{AL}$ language is a restriction of $\mathcal{ALC}$ without union of concept ($\sqcup$), negation is only allowed to atomic concepts and limited existential quantification, that is, existential quantification only over $\top$ concept ($\exists R.\top$). The $\mathcal{U}$ stands for union of concepts, $\mathcal{E}$ for full existential quantification, $\mathcal{N}$ for number restrictions, $\mathcal{Q}$ for qualified number restrictions and $\mathcal{C}$ for full negation of concepts (not only atomics ones).

Taking into account the semantics, some of these languages are equivalent. For example, the semantics forces the equivalent between $C \sqcup D \equiv \neg(\neg C \sqcap \neg D)$ and $\exists R.C \equiv \neg\forall R.\neg C$. That is, union and full existential quantification can be expressed using negation and vice versa. That is why we use $\mathcal{ALC}$ instead of $\mathcal{ALUE}$ and $\mathcal{ALCN}$ instead of $\mathcal{ALUEN}$. One can also observe that $\mathcal{ALCN}$ is superseded by $\mathcal{ALCQ}$. That is, if we limit the qualified number restrictions of $\mathcal{ALCQ}$ to the $\top$ concept allowing only $\leq nR.\top$ and $\geq nR.\top$, we obtain $\mathcal{ALCN}$.

As said before, description logics are a huge family of formalisms. Much more constructors were introduced in the basic $\mathcal{ALC}$ to express: role constructors; concrete domains; modal, epistemic and temporal operators; *fuzzy* and probabilities to express uncertain or vague knowledge to cite just some of them [1]. Nevertheless, the languages presented so far will be sufficient for us in this work.

[1] We also point to the Description logics website at http://dl.kr.org/.

2.4 Reasoning in DLs

A knowledge base—TBox and ABox—equipped with its semantics is equivalent to a set of axioms in first-order predicate logic. Thus, as said before, like any other set of axioms, it contains implicit knowledge that *logical inferences* can make explicit.

When we are constructing a TBox $\mathcal{T}$, by defining new concepts, possibly in terms of others that have been defined before, it is important to enforce the consistence of the TBox. That is, it is important that new concepts make sense and do not be contradictory with old ones. Formally, a concept makes sense if there is some interpretation that satisfies the axioms of $\mathcal{T}$ such that the concept denotes a nonempty set in that interpretation.

Definition 5 (*Satisfiability*) A concept C is *satisfiable* with respect to $\mathcal{T}$ if there is a model $\cdot^{\mathcal{I}}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In this case, $\cdot^{\mathcal{I}}$ is a *model* of C.

While modeling a domain of knowledge into a TBox other important inference service is necessary. For instance, it is usually interesting to organize the concepts of a TBox into a taxonomy. That is, it is important to know whether some concept is more general than another one: the *subsumption problem*. Furthermore, other interesting relationships among concepts is the *equivalence*.

Definition 6 (*Subsumption*) A concept C is *subsumed* by a concept D with respect to $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\cdot^{\mathcal{I}}$ of $\mathcal{T}$. In this case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

Definition 7 (*Equivalence*) Two concepts C and D are *equivalent* with respect to $\mathcal{T}$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\cdot^{\mathcal{I}}$ of $\mathcal{T}$. In this case we write $C \equiv_{\mathcal{T}} D$ or $\mathcal{T} \models C \equiv D$.

If the TBox is clear from the context or empty we can drop the qualification and simply write $\models C \sqsubseteq D$ if C is subsumed by D, and $\models C \equiv D$ if they are equivalent.

Since it is not our main concern in this book, we will not go into more details about the equivalence and reductions between reasoning problems in Description Logics. Basically, the different kinds of reasoning can be all reduced to a main inference problem named the consistency check for ABox [1].

2.5 Inference Algorithms

There are two main algorithms to reasoning in Description Logics: *structural subsumption algorithms* and *tableaux-based algorithms* [1]. We postpone the presentation of these two algorithms for Chap. 4, here we will just present briefly comments about them. One of the differences between them relies on the logical languages that each one can handle.

For the description logic $\mathcal{ALN}$ and its subsets, that is, the Description Logic not allowing full negation ($\neg C$), disjunction ($C \sqcup D$) nor full existential ($\exists R.C$), the subsumption of concepts can be computed by structural subsumption algorithms. The idea of these algorithms is compare the syntactic structure of concept descriptions. These algorithms are usually very efficient, polynomial time complexity [2] indeed.

For $\mathcal{ALC}$ and its extensions, the satisfiability of concepts and the subsumption of concept usually can be computed by *tableau-based algorithms* which are sound and complete for these problems [1]. The first tableau-based algorithm for satisfiability of $\mathcal{ALC}$-concepts was presented by [4]. As we said before, some reasoning problems in Description Logics can be reduced to others, in special, the problem to test the subsumption of concepts is reduced to the problem of test the (un)satisfiability of a concept description. These algorithms use the fact that $C \sqsubseteq D$ if and only if $C \sqcap \neg D$ is unsatisfiable [1]. Regarding the complexity, the tableau-based satisfiability algorithm for $\mathcal{ALC}$ is a PSPACE-hard problem [4].

2.6 $\mathcal{ALC}$ Axiomatization

From [3] we know that $\mathcal{ALC}$ is sound and complete for any Classical Propositional Logic axiomatization containing the axioms:

Definition 8 (*An Axiomatization of $\mathcal{ALC}$*)

$$\forall R.(\alpha \sqcap \beta) \equiv \forall R.\alpha \sqcap \forall R.\beta \tag{2.1}$$

$$\forall R.\top \equiv \top \tag{2.2}$$

As usual, $\exists R.\alpha$ can be taken as a shorthand for $\neg\forall R.\neg\alpha$, as well as $\forall R.\alpha$ as a shorthand for $\neg\exists R.\neg\alpha$. Taking $\exists R.\alpha$ as a definable concept, the axioms change to

$$\exists R.(\alpha \sqcup \beta) \equiv \exists R.\alpha \sqcup \exists R.\beta \tag{2.3}$$

$$\exists R.\bot \equiv \bot \tag{2.4}$$

The following rule, also known as necessitation rule:

$$\frac{\vdash \alpha}{\vdash \forall R.\alpha}\ Nec$$

is sound and complete for $\mathcal{ALC}$ semantics. In fact, by Lemmas 1 and 2, the Axiom 2.1 and this necessitation rule are an alternative axiomatization for $\mathcal{ALC}$.

Lemma 1 *The necessitation rule is a derived rule in the above Axiomatization.*

Proof Let α be a tautology, so that, from $\mathcal{ALC}$ semantics, $\alpha \equiv \top$ and hence, $\forall R.\alpha \equiv \forall R.\top$. Now, from the Axiom $\forall R.\top \equiv \top$ and we can conclude that $\forall R.\alpha \equiv \top$, that is, $\forall R.\alpha$ is also a tautology, and so it is provable by completeness. □

Lemma 2 *The Axiom $\forall R.\top \equiv \top$ is derived from the necessitation rule.*

Proof If the necessitation rule is valid then whenever its premise is valid, its conclusion is valid. $\top$ is provable, so we can conclude that $\forall R.\top$ is also provable by the necessitation rule, so the equivalence $\forall R.\top \equiv \top$ is also provable.

Finally, we can state two useful facts following directly from the $\mathcal{ALC}$ semantics. Those facts will be used during this book to prove the soundness of the presented deduction systems. □

Fact 1 *If* $C \sqsubseteq D$ *then* $\exists R.C \sqsubseteq \exists R.D$.

Fact 2 *If* $C \sqsubseteq D$ *then* $\forall R.C \sqsubseteq \forall R.D$.

References

1. Baader, F.: The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Levesque, H., Brachman, R.: Expressiveness and tractability in knowledge representation and reasoning. Computat. Intell. **3**(2), 78–93 (1987)
3. Schild, K.: A correspondence theory for terminological logics: preliminary report. In: IJCAI'91: Proceedings of the 12th International Joint Conference on Artificial Intelligence, pp. 466–471. Morgan Kaufmann Publishers Inc., San Francisco, (1991). Also published at TR 91, Technische Universitat Berlin
4. Schmidt-Schauß, M., Smolka, G.: Attributive concept descriptions with complements. Artif. Intell. **48**(1), 1–26 (1991)