

DESCRIPTION LOGIC

Franz Baader and Carsten Lutz

1	Introduction	757
2	Basic Definitions and Connection to Modal Logic	759
	2.1 Concept Languages	759
	2.2 Terminological Formalisms	763
	2.3 Assertional Formalisms	766
3	Standard Description Logic Inferences	767
	3.1 Terminological Reasoning	767
	3.2 Assertional Reasoning	770
	3.3 Compound Inference Problems	771
4	Sub-Boolean Description Logics	773
	4.1 Reasoning with concept descriptions in sub-Boolean DLs	775
	4.2 TBox reasoning in sub-Boolean DLs	782
	4.3 ABox reasoning in sub-Boolean DLs	786
	4.4 Bi-simulation characterizations of sub-Boolean DLs	787
5	Non-Standard Inferences	788
	5.1 Motivation	789
	5.2 Least common subsumers and most specific concepts	791
	5.3 Matching and unification	793
	5.4 Rewriting and approximation	798
6	Non-Standard Expressivity	801
	6.1 Concrete Domains	801
	6.2 Role Value Maps	808

1 INTRODUCTION

Description logics (DLs) [12] are a family of knowledge representation languages which can be used to represent the terminological knowledge of an application domain in a structured and formally well-understood way. The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are described by *concept descriptions*, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. For example, the concept of “a man that is married to a doctor, and has only happy children” can be expressed using the concept description

$$\text{Man} \sqcap \exists \text{married}.\text{Doctor} \sqcap \forall \text{child}.\text{Happy}.$$

On the other hand, DLs differ from their predecessors in that they are equipped with a formal, *logic*-based semantics, which can, e.g., be given by a translation into first-order predicate logic. For example, the above concept description can be translated into the following first-order formula (with one free variable x):

$$\text{Man}(x) \wedge \exists y(\text{married}(x, y) \wedge \text{Doctor}(y)) \wedge \forall y(\text{child}(x, y) \rightarrow \text{Happy}(y)).$$

The motivation for introducing the early predecessors of DLs, such as semantic networks and frames [133, 125], actually was to develop means of representation that are closer to the way humans represent knowledge than a representation in formal logics, like first-order predicate logic. Minsky [125] even combined his introduction of the frame idea with a general rejection of logic as an appropriate formalism for representing knowledge. However, once people tried to equip these “formalisms” with a formal semantics, it turned out that they can be seen as syntactic variants of (subclasses of) first-order predicate logic [83, 144].

The immediate precursors of DLs, Brachman’s structured inheritance networks [42], were an attempt to define a formalism that allows for a structured representation of knowledge in the spirit of semantics networks and frames, but nevertheless is equipped with a formal semantics. The original description logics used in systems that implemented these ideas in the 1980ies [45, 132, 124, 123] turned out to correspond to rather inexpressive and somewhat unusual subclasses of first-order predicate logic. On the one hand, none of them was propositionally closed since they did not allow for disjunction or negation. On the other hand, they were equipped with certain other complex constructors (like number restrictions and role-value-maps), which, though expressible in first-order predicate logic, are not considered as atomic constructors there. For example, the number restriction (≥ 5 child) describes people having at least five children, and the role-value-map $\text{child} \circ \text{friend} \subseteq \text{know}$ describes people that know all their children’s friends.

The main inference problem to be solved in description logics is the subsumption problem, i.e., deciding whether one concept is a subconcept of another one. The early DL systems cited above employed so-called structural subsumption algorithms, which first normalise the concept descriptions, and then recursively compare the syntactic structure of the normalised descriptions. These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for rather inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption relationships. To overcome this problem, Schmidt-Schauß and Smolka [143] made DLs into “real” logics by introducing negation. Their main motivation for this was that they wanted to reduce the subsumption problem to the satisfiability problem. They introduced a basic propositionally closed DL, which they called \mathcal{ALC} , developed a tableau-like algorithm for satisfiability in \mathcal{ALC} , and showed that the subsumption and satisfiability problem in \mathcal{ALC} are PSPACE-complete.

A reader of the Handbook of Modal Logic who followed us so far may rightfully ask: *And what has all this to do with Modal Logic?* The answer was given by Schild, who noticed that \mathcal{ALC} is just a syntactic variant of multi-modal K, i.e., the basic modal logic of Kripke frames with several accessibility relations (and thus several pairs of box- and diamond operators). In fact, the translations of \mathcal{ALC} and of K into first-order predicate logic yield exactly the same class of first-order formulae. This connection between DLs and modal logic was used by Schild and others (see, e.g., [139, 140, 54, 55]) to transfer decidability and complexity results from modal logic to DLs, but also to extend these

results to logics with other DL constructors. At the same time, tableau-based algorithms were developed for more and more expressive DLs (see [30] for an overview), and highly-optimized implementations of these algorithms [92] turned out to behave quite well on artificial benchmarks from modal logic [131] and also in practice [78].

Though there is a very close connection between DLs and modal logics (MLs), the underlying intuition as well as the intended applications differ significantly. As a consequence, the focus of research in DL and in modal logic also differs. While mentioning the similarities, this chapter will focus on topics that are specific for DLs.

Section 2 formally introduces syntax and semantics of the basic DL \mathcal{ALC} , and shows its relationship to multi-modal \mathbf{K} . It then introduces additional DL constructors, and describes their ML counterparts. In addition to these constructors, DLs provide their users with a terminological formalism, which (in its simplest form) allows to introduce names for complex concepts, and an assertional formalism, which allows to state facts about specific individuals/objects. Though these components are usually not available in ML, there are some connections to things known in ML (such as nominals, the universal modality, fixpoint operators, etc.).

In *Section 3*, we introduce the standard inference problems in description logics, show how they can be reduced to each other, and how they relate to inference problems in ML. The standard way of solving these problems in propositionally closed DLs is using tableau-based algorithms. Since these algorithms are treated in other chapters, we only give some references to the relevant chapters.

Section 4 considers DLs that are not propositionally closed, and where consequently subsumption cannot be reduced to satisfiability. We review the known complexity results for such DLs, and then describe (complete) structural subsumption algorithms for some of them. In addition, we mention bi-simulation characterizations of the corresponding ML fragments.

Section 5 is concerned with so-called non-standard inferences in DLs, like computing the least common subsumer and the most specific concept, and rewriting, unification, and matching of concepts. These inferences have been introduced with the goal of supporting the user when building and maintaining large DL knowledge bases. With the exception of unification, none of them have been investigated in ML.

Finally, *Section 6* introduces means of expressiveness that do not have immediate ML counterparts.

2 BASIC DEFINITIONS AND CONNECTION TO MODAL LOGIC

In this section, we introduce the basic components of description logics: concept languages, terminological formalisms, and assertional formalisms.

2.1 Concept Languages

We first define the basic propositionally closed concept language \mathcal{ALC} introduced by Schmidt-Schauß and Smolka [143], and then describe a number of natural extensions that are important for many applications and offered by modern DL reasoners. Assume that a countably infinite supply of *concept names*, usually denoted A and B , and of *role names*, usually denoted r and s , are available. *Concept descriptions* in \mathcal{ALC} are formed