1
Introduction

This is, to the best of our knowledge, the first textbook dedicated solely to Description Logic (DL), a very active research area in logic-based knowledge representation and reasoning that goes back to the late 1980s and that has a wide range of applications in knowledge-intensive information systems. In this introductory chapter we will sketch what DLs are, how they are used and where they come from historically. We will also explain how to use this book.

1.1 What are DLs and where do they come from?

Description logics (DLs) are a family of knowledge representation languages that can be used to represent knowledge of an application domain in a structured and well-understood way.[1] The name *description logics* is motivated by the fact that, on the one hand, the important notions of the domain are represented 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; on the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a *logic*-based semantics which, up to some differences in notation, is actually the same semantics as that of classical first-order logic.

Description logics typically separate domain knowledge into two components, a *terminological* part called the TBox and an *assertional* part called the ABox, with the combination of a TBox and an ABox being called a *knowledge base* (KB). The TBox represents knowledge about the structure of the domain (similar to a database schema), while the ABox represents knowledge about a concrete situation (similar to a database

[1] Note that we use *Description Logic* (singular) to refer to the research area, and *description logics* (plural) to refer to the relevant logical formalisms.

instance). TBox statements capturing knowledge about a university domain might include, e.g., *a teacher is a person who teaches a course*, *a student is a person who attends a course* and *students do not teach*, while ABox statements from the same domain might include *Mary is a person*, *CS600 is a course* and *Mary teaches CS600*. As already mentioned, a crucial feature of DLs is that such statements have a formal, logic-based semantics. In fact the above statements can be rendered as sentences in first-order logic as follows:

$$\begin{array}{l}\forall x\,(\mathsf{Teacher}(x) \Leftrightarrow \mathsf{Person}(x) \wedge \exists y\,(\mathit{teaches}(x,y) \wedge \mathsf{Course}(y))),\\ \forall x\,(\mathsf{Student}(x) \Leftrightarrow \mathsf{Person}(x) \wedge \exists y\,(\mathit{attends}(x,y) \wedge \mathsf{Course}(y))),\\ \forall x\,((\exists y\,\mathit{teaches}(x,y)) \Rightarrow \neg\mathsf{Student}(x)),\\ \mathsf{Person}(\mathsf{Mary}),\\ \mathsf{Course}(\mathsf{CS600}),\\ \mathit{teaches}(\mathsf{Mary},\mathsf{CS600}).\end{array}$$

Equivalently, these statements can be written in description logic syntax as follows:

$$\begin{array}{l}\mathsf{Teacher} \equiv \mathsf{Person} \sqcap \exists \mathit{teaches}.\mathsf{Course},\\ \mathsf{Student} \equiv \mathsf{Person} \sqcap \exists \mathit{attends}.\mathsf{Course},\\ \exists \mathit{attends}.\top \sqsubseteq \neg\mathsf{Student},\\ \mathsf{Mary} : \mathsf{Person},\\ \mathsf{CS600} : \mathsf{Course},\\ (\mathsf{Mary},\mathsf{CS600}) : \mathit{teaches}.\end{array}$$

The first three statements of this knowledge base constitute its TBox, and the last three its ABox. Please note how the DL syntax does not use variables x or y. In Chapter 2 an extended version of the university KB example will be used to define and explain DL syntax and semantics in detail.

The logic-based semantics of DLs means that we have a well-defined, shared understanding of when a statement is *entailed* by a KB; for example, the above KB entails that Mary is a teacher. Moreover, we can use (automated) reasoning to determine those entailments, and thus reasoning can be used to support the development and application of DL KBs. Common reasoning tasks include checking the satisfiability of concepts and the consistency of KBs, determining when one concept is more specific than another (a reasoning task called *subsumption*) and answering different kinds of database-style queries over the KB.

The power of DLs derives from the fact that reasoning tasks are performed with respect to the whole KB, and in particular with respect to the conceptual domain knowledge captured in the TBox. Unfortunately, this power does not come without a computational cost, and one of the most important areas of DL research has been exploring the trade-off between the expressive power of the language available for making statements (particularly TBox statements) and the computational complexity of various reasoning tasks. The expressive power of DLs is invariably constrained so as to at least ensure that common reasoning tasks are decidable (i.e., they can always be correctly completed in a finite amount of time), and may even be sufficiently constrained so as to make them tractable (i.e., they can always be correctly completed in time that is polynomial with respect to the size of the KB). In another area of DL research, its *model theory*, we investigate which kinds of semantic structure, i.e., interpretations or models, we can describe in a KB. As well as theoretical investigations, e.g., determining the worst-case complexities for various DLs and reasoning problems, there has also been extensive practical work, e.g., developing systems and optimisation techniques, and empirically evaluating their behaviour when applied to benchmarks or KBs used in various applications. We will explore model theory in Chapter 3, theoretical complexity issues in Chapter 5 and DL reasoning techniques in Chapters 4, 6 and 7.

The emphasis on decidable and tractable formalisms is also the reason why a great variety of extensions of basic DLs have been considered – combining different extensions can easily lead to undecidability or intractability, even if each of the extensions is harmless when considered in isolation. While most DLs can be seen as decidable fragments of first-order logic, some extensions leave the realm of classical first-order logic, including, e.g., DLs with modal and temporal operators, fuzzy DLs and probabilistic DLs (for details, see [BCM$^+$07, Chapter 6] and specialised survey articles such as [LWZ08, LS08]). If an application requires more expressive power than can be provided by a decidable DL, then one usually embeds the DL into an application program or another KR formalism rather than using an undecidable DL.

1.2 What are they good for and how are they used?

DL systems have been used in a range of application domains, including configuration (e.g., of telecommunications equipment) [MW98], software information and documentation systems [DBSB91] and databases [BCM$^+$07], where they have been used to support schema design [CLN98, BCDG01], schema and data integration [CDGL$^+$98b, CDGR99], and query answering [CDGL98a, CDGL99, HSTT00]. More recently, DLs have played a central role in the semantic web [Hor08], where they have been adopted as the basis for ontology languages such as OWL [HPSvH03], and its predecessors OIL and DAML+OIL, and DL knowledge bases are now often referred to as ontologies. This has resulted in a more widespread use of DL systems, with applications in fields as diverse as agriculture [SLL$^+$04], astronomy [DeRP06], biology [RB11, OSRM$^+$12], defence [LAF$^+$05], education [CBV$^+$14], energy management [CGH$^+$13], geography [Goo05], geoscience [RP05], medicine [CSG05, GZB06, HDG12, TNNM13], oceanography [KHJ$^+$15b] and oil and gas [SLH13, KHJ$^+$15a].

In a typical application, the first step will be to determine the relevant vocabulary of the application domain and then formalise it in a suitable TBox. This *ontology engineering* process may be manual or (semi-)automatic. In either case a DL reasoner is invariably used to check satisfiability of concepts and consistency of the ontology as a whole. This reasoner is often integrated in an ontology editing tool such as Protégé [KFNM04]. Some applications use only a terminological ontology (i.e., a TBox), but in others the ontology is subsequently used to structure and access data in an ABox or even in a database. In the latter case a DL reasoner will again be used to compute query answers.

The use of DLs in applications throws the above mentioned expressivity versus complexity trade-off into sharp relief. On the one hand, using a very restricted DL might make it difficult to precisely describe the concepts needed in the ontology and forces the modelling to remain at a high level of abstraction; on the other hand, using a highly expressive DL might make it difficult to perform relevant reasoning tasks in a reasonable amount of time. The OWL ontology language is highly expressive, and hence also highly intractable; however, the currently used OWL 2 version of OWL also specifies several *profiles*, fragments of the language that are based on less expressive but tractable DLs. We will discuss OWL and OWL 2 in more detail in Chapter 8.

1.3 A brief history of description logic

The study of description logic grew out of research into knowledge representation systems, such as semantic networks and frames, and a desire to provide them with precise semantics and well-defined reasoning procedures [WS92]. Early work was mainly concerned with the implementation of systems, such as KL-ONE, K-REP, BACK, and LOOM [BS85, MDW91, Pel91, Mac91a]. These systems employed so-called *structural subsumption algorithms*, which first normalise the concept descriptions, and then recursively compare the syntactic structure of the normalised descriptions [Neb90a]. These algorithms are usually relatively efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot derive all relevant entailments. Early formal investigations into the complexity of reasoning in DLs showed that most DLs do not have polynomial-time inference problems [BL84, Neb90b]. Influenced by these results, the implementors of the CLASSIC system (the first industrial-strength DL system) chose to carefully restrict the expressive power of their DL so as to allow for tractable and complete reasoning [PSMB$^+$91, Bra92].

The so-called *tableau* reasoning technique for DLs was first introduced by Schmidt-Schauß and Smolka in the early 1990s [SS91]. Tableau algorithms work on propositionally closed DLs (i.e., DLs with full Boolean operators), and are complete even for very expressive DLs. Moreover, an implementation of one such algorithm in the KRIS system showed that, with suitable optimisations, performance on realistic problems could be comparable with or even superior to existing structural approaches [BFH$^+$92]. At the same time, there was a thorough analysis of the complexity of reasoning in various DLs [DLNN91a, DLNN91b, DHL$^+$92], and it was observed that DLs are very closely related to modal logics [Sch91].

Initially, tableau algorithms and systems, including KRIS, considered only relatively restricted DLs (see Section 4.2.2). On the theoretical side, tableau algorithms were soon extended to deal with more expressive DLs [HB91, Baa91, BH91, BDS93]. It took several years, however, before the FaCT system demonstrated that suitably optimised implementations of such algorithms could be effective in practice [Hor97]. Subsequently, tableau algorithms were developed for increasingly expressive DLs [HST00], and implemented in FaCT and in other highly optimised DL systems including RACER [HM01], FaCT++ [TH06] and Pellet [SPC$^+$07]. This line of research culminated in the development of $\mathcal{SROIQ}$ [HKS06], the DL that forms the basis for the OWL ontology language. In fact, a DL knowledge base can be seen as an OWL ontology. The standardisation of OWL gave DLs a stable, machine-processable and web-friendly syntax; this, and the central role of ontologies in the semantic web, sparked an increased development of DL knowledge bases (and OWL ontologies), and an increased development effort for tools such as reasoners to determine entailments, ontology editors to write knowledge bases and APIs to programmatically access ontologies and reasoners (see Section 8.2).

During the same period, the relationship to modal logics [DGL94a, Sch95] and to decidable fragments of first-order logic was also studied in more detail [Bor96, PST97, GKV97, Grä98, Grä99, LSW01], and first applications in databases (such as schema reasoning, query optimisation, and data integration) were investigated [LR96, BDNS98, CDGL98a, CDGL$^+$98b].

Although highly optimised implementations of tableau algorithms were successful in many TBox reasoning applications, some larger-scale ontologies proved stubbornly resistant. Moreover, it remained unclear how tableau reasoning could deal effectively with very large ABoxes. This revived the interest in less expressive DLs, with the goal of developing tools that can deal with very large TBoxes and/or ABoxes, and led to the development of the $\mathcal{EL}$ and DL-Lite families of tractable DLs [BBL05, BBL08, CGL$^+$05, CDL$^+$07, ACKZ09], which are both included in OWL 2 as profiles. A main advantage of the $\mathcal{EL}$ family is that it is amenable to *consequence-based* reasoning techniques which scale also to large ontologies and are more robust than tableau reasoning [BBL05]. This was first demonstrated by the CEL system [BLS06]; other relevant implementations include ELK [KKS14] and SnoRocket [MJL13].

With the advent of the DL-Lite family of DLs, applications of description logics in databases started to receive increased interest. There are various benefits to enriching a database application with an ontology, such as adding domain knowledge, giving a formal definition to the symbols used in the database and providing an enriched and unified schema that can be used to formulate queries. These ideas have led to the study of *ontology-mediated querying* [BtCLW14] and to the *ontology-based data access (OBDA)* paradigm for data integration [CDL$^+$09]; see also the recent surveys [KZ14, BO15]. DL-Lite is particularly suitable for such applications since its expressive power is sufficiently restricted so that database-style query answering with respect to ontologies can be reduced via query rewriting techniques to query answering in relational databases (see Chapter 7); this in turn allows standard database systems to be used for query answering in the presence of ontologies [CDL$^+$07]. Implemented systems in this area include QuOnto and Mastro [ACG$^+$05, CCD$^+$13] as well as Ontop [KRR$^+$14].

As DLs became increasingly used, researchers investigated a multitude of additional reasoning tasks that are intended to make DLs more usable in various applications. These included, among many others, computing *least common subsumers* and *concept difference*, *ontology difference*, and *explanation* [BK06, KWW08, HPS09]. The need to support the modularity of ontologies has been a strong driving force for studying new reasoning problems such as module extraction [GHKS08], conservative extensions [GLW06], and inseparability [BKL$^+$16]. These tasks are now widely used to support ontology engineering, and so is explanation: module extraction and inseparability can be used to support ontology reuse, e.g., by highlighting interactions between statements in different ontologies, and explanation can be used to help debug errors in ontologies, e.g., by highlighting the causes of inconsistencies.

Description Logic continues to be a very active research area, with new theoretical results and new reasoning techniques and systems constantly being developed; see `http://dl.kr.org/`. These include the extension of tableau to hypertableau, as implemented in the HermiT system [GHM$^+$14], the extension of rewriting techniques to the $\mathcal{EL}$ family of DLs and beyond [PUMH10, LTW09, BLW13, SMH13, BtCLW14], as implemented in the KARMA [SMH13] and Grind [HLSW15] systems, and the development of hybrid techniques, e.g., combining tableau with consequence-based approaches in the Konclude system [SLG14].

1.4 How to use this book

This book is intended as a textbook and not as a research monograph. Consequently, we have tried to cover all core aspects of DLs at a level of detail suitable for a novice reader with a little background in formal methods or logic. In particular, we expect the reader to understand the basic notions around sets, relations and functions, e.g., their union, intersection or composition. It will be useful, but not essential, for readers to have some knowledge of first-order logic and basic notions from theoretical computer science. Those lacking such background may wish to consult appropriate textbooks, e.g., `http://phil.gu.se/logic/books/Gallier:Logic_For_Computer_Science.pdf` (which also contains a nice example of a guide for readers).

This book includes both basic and advanced level material suitable for undergraduate through to introductory graduate level courses on description logics. In the authors' experience, the material included here could be covered in a 36-hour lecture course for students with a good background in logic. For shorter courses, or those aimed at a different cohort, some of the more advanced material can easily be dropped.

Chapters 2 and 3 provide background material, including examples and definitions, that will prove useful in the remaining chapters. Some parts of these chapters are, however, quite long and detailed, and it may not be appropriate to read (or teach) them in full before continuing with the remainder of the book, but rather to dip into them as need arises. Also, the subsequent chapters are presented in an order that the authors find didactically convenient, but the order in which they are read and/or taught could easily be varied.

Chapter 4 deals with tableau-based reasoning techniques; these are typically used to reason about expressive DLs. It presents tableau algorithms for ABox and KB consistency in the basic DL $\mathcal{ALC}$, and shows how they can be extended to deal with other concept and role constructors. The chapter also includes a brief discussion of implementation issues. Chapter 5 discusses the computational complexity of satisfiability and subsumption in a variety of expressive DLs, and proves upper and lower complexity bounds for a suitable set of these problems. It also gives examples of extensions of DLs that are too expressive in the sense that they lead to undecidability. Chapter 6 looks at reasoning in the inexpressive DL $\mathcal{EL}$ and explains the consequence-based reasoning technique for this logic, and it also showcases an extension (with inverse roles) in which reasoning is more challenging. So far in this book, reasoning has been restricted to determining whether a DL knowledge base entails a DL axiom. Chapter 7 discusses more complex reasoning problems, namely query answering: the entailments to be checked are from a different language, in particular conjunctive queries and first-order queries. Finally, Chapter 8 explains the relationship between OWL and DLs, and describes the tools and applications of OWL.

In Chapters 2–7, citations have been kept to a minimum, but most chapters conclude with a short section providing historical context and a literature review.

The reader is cordially invited to *actively* read this book, especially the basic definitions. Throughout the book, we provide a lot of examples but strongly suggest that, whenever a new notion or term is introduced, the reader should consider their own examples of this notion or term – possibly by varying the ones presented – in order to make sure that the newly introduced notion is completely understood. We also show how to draw interpretations and models, and explain reasoning algorithms. Again, in addition to the examples given, the reader should draw their own models and run the algorithms on other inputs.

The running teaching example used throughout this book is made available on the book's website at `http://dltextbook.org/` in an OWL syntax. You will also find useful further examples and exercises there, as well as a list of errata, to which you can contribute by informing us about any errors that you find in the book.