Mainstream DL research of the last 25 years: towards very expressive DLs with practical inference procedures

Description Logics [5] are a well-investigated family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, and databases, but their most notable success so far is the adoption of the DL-based language OWL (http://www.w3.org/TR/owl-features/) as the standard ontology language for the Semantic Web [37].

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, that is expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using concept constructors. The expressivity of a particular DL is determined by which concept constructors are available in it. From a semantic point of view, concept names and concept descriptions represent sets of individuals, whereas roles represent binary relations between individuals. For example, using the concept names Man, Doctor, and Happy and the role names married and child, the concept of “a man that is married to a doctor, and has only happy children” can be expressed using the concept description

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

On the other hand, DLs differ from their predecessors in that they are equipped with a formal, logic-based semantics, which can, for example 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 \( x \)):

\[
\text{Man}(x) \land \exists y. (\text{married}(x, y) \land \text{Doctor}(y)) \\
\land \forall y. (\text{child}(x, y) \rightarrow \text{Happy}(y)).
\]

The motivation for introducing the early predecessors of DLs, such as semantic networks and frames [50, 57], 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 [50] 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 [33, 65]. Description Logics were developed with the intention of keeping the advantages of the logic-based approach to knowledge representation (like a formal model-theoretic semantics and well-defined inference problems), while avoiding the disadvantages of using full first-order predicate logic (e.g., by using a variable-free syntax that is easier to read, and by ensuring decidability of the important inference problems).
Abstract

Mainstream research in Description Logics (DLs) until recently concentrated on increasing the expressive power of the employed description language while keeping standard inference problems like subsumption and instance manageable in the sense that highly-optimized reasoning procedure for them behave well in practice. One of the main successes of this line of research was the adoption of OWL DL, which is based on an expressive DL, as the standard ontology language for the Semantic Web.

More recently, there has been a growing interest in more light-weight DLs, and in other kinds of inference problems, mainly triggered by need in applications with large-scale ontologies. In this paper, we first review the DL research leading to the very expressive DLs with practical inference procedures underlying OWL, and then sketch the recent development of light-weight DLs and novel inference procedures.

Concept descriptions can be used to define the terminology of the application domain, and to make statements about a specific application situation in the assertional part of the knowledge base. In its simplest form, a DL terminology (usually called TBox) can be used to introduce abbreviations for complex concept descriptions. For example, the concept definitions

\[
\begin{align*}
\text{Man} & \equiv \text{Human} \sqcap \neg \text{Female}, \\
\text{Woman} & \equiv \text{Human} \sqcap \text{Female}, \\
\text{Father} & \equiv \text{Man} \sqcap \exists \text{child. } \top
\end{align*}
\]

define the concept of a man (woman) as a human that is not female (is female), and the concept of a father as a man that has a child, where \( \top \) stands for the top concept (which is interpreted as the universe of all individuals in the application domain). The above is a (very simple) example of an acyclic TBox, which is a finite set of concept definitions that is unambiguous (i.e., every concept name appears at most once on the left-hand side of a definition) and acyclic (i.e., there are no cyclic dependencies between definitions). In general TBoxes, so-called general concept inclusions (GCIs) can be used to state additional constraints on the interpretation of concepts and roles. In our example, it makes sense to state domain and range restrictions for the role child. The GCIs

\[
\exists \text{child. } \text{Human} \sqsubseteq \text{Human} \quad \text{and} \\
\text{Human} \sqsubseteq \forall \text{child. } \text{Human}
\]

say that only human beings can have human children, and that the child of a human being must be human.

In the assertional part (ABox) of a DL knowledge base, facts about a specific application situation can be stated by introducing named individuals and relating them to concepts and roles. For example, the assertions

\[
\text{Man(\text{JOHN})}, \quad \text{child(\text{JOHN, MACKENZIE})}, \\
\text{Female(\text{MACKENZIE})},
\]

state that John is a man, who has the female child Mackenzie.

Knowledge representation systems based on DLs provide their users with various inference services that allow them to deduce implicit knowledge from the explicitly represented knowledge. For instance, the subsumption algorithm allows one to determine subconcept-superconcept relationships. For example, w.r.t. the concept definitions from above, the concept Human subsumes the concept Father since all instances of the second concept are necessarily instances of the first concept, that is whenever the above concept definitions are satisfied, then Father is interpreted as a subset of Human. With the help of the subsumption algorithm, one can compute the hierarchy of all concepts defined in a TBox. This inference service is usually called classification. The instance algorithm can be used to check whether an individual occurring in an ABox is necessarily an instance of a given concept. For example, w.r.t. the above assertions, concept definitions, and GCIs, the individual MACKENZIE is an instance of the concept Human. With the help of the instance algorithm, one can compute answers to instance queries, that is all individuals occurring in the ABox that are instances of the query concept \( C \).

In order to ensure a reasonable and predictable behavior of a DL system, the underlying inference problems (like the subsumption and the instance problem) should at least be decidable for the DL employed by the system, and preferably of low complexity. Consequently, the expressive power of the