Description Logics for the Semantic Web

Franz Baader, Ian Horrocks, Ulrike Sattler

November 7, 2001

Abstract

The vision of a Semantic Web has recently drawn considerable attention, both from academia and industry. Description Logics are often named as one of the tools that can support the Semantic Web and thus help to make this vision reality.

In this paper, we sketch what Description Logics are and what they can do for the Semantic Web. Descriptions Logics are very useful for defining, integrating, and maintaining ontologies, which provide the Semantic Web with a common understanding of the basic semantic concepts used to annotate Web pages. We also argue that, without the last decade of basic research in this area, Description Logics could not play such an important rôle in this domain.

The Semantic Web and Ontologies

For many people, the World Wide Web has become an indispensable means of providing and searching for information. Searching the Web in its current form is, however, often an infuriating experience since today's search engines usually provide a huge number of answers, many of which are completely irrelevant, whereas some of the more interesting answers are not found. One of the reasons for this unsatisfactory state of affairs is that existing Web resources are usually only human understandable: the mark-up (HTML) only provides rendering information for textual and graphical information intended for human consumption.

The Semantic Web [1] aims for machine-understandable Web resources, whose information can then be shared and processed both by automated tools, such as search engines, and by human users. In the following we will refer to consumers of Web resources, whether automated tools or human users, as agents. This sharing of information between different agents requires semantic mark-up, i.e., an annotation of the Web page with information on its content that is understood by the agents searching the Web. To make sure that different agents have a common understanding of this semantic mark-up, one needs ontologies that establish a joint terminology between the agents. Basically, an ontology [**?**] is a collection of definitions of concepts and the shared understanding comes from the fact that all the agents interpret the concepts w.r.t. the same ontology.

The use of ontologies in this context requires a well-designed, well-defined, and Web-compatible ontology language with supporting reasoning tools. The syntax of this language should be both intuitive to human users and compatible with existing Web standards (such as XML, RDF and RDFS). Its semantics should be formally specified since otherwise it could not provide a shared understanding. Finally, its expressive power should be adequate, i.e., the language should be expressive enough for defining the relevant concepts in enough detail, but not too expressive to make reasoning infeasible. Reasoning is important to ensure the quality of an ontology. It can be employed in different development phases. During ontology design, it can be used to test whether concepts are non-contradictory and to derive implied relations. In particular, one usually wants to compute the concept hierarchy since information on which concept is a specialization of another and which concepts are synonyms is very useful when searching Web pages annotated with such concepts. Since it is not reasonable to assume that there will be a single ontology for the whole Web, interoperability and integration of different ontologies is also an important issue. Integration can, for example, be supported by asserting inter-ontology relationships and testing for consistency and computing the integrated concept hierarchy. Finally, reasoning may also be used when the ontology is deployed, i.e., with Web pages that are already annotated with its concepts. One can, for example, determine the consistency of facts stated in the annotation with the ontology or infer instance relationships. However, in the deployment phase, the requirements on the efficiency of reasoning are much more stringent than in the design and integration phases.

Before arguing why Description Logics are good candidates for such an ontology language, we provide a brief (and informal) introduction to and history of Description Logics.

Description Logics

Description logics (DLs) [**?**, **?**] are a family of knowledge representation languages that can be used to represent the 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. On the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, *logic*-based semantics.

Instead of giving a formal definition of concept constructors and their semantics, we illustrate some typical constructors by an example. Formal definitions can, e.g., be found in [**?**, **?**]. Assume that we want to define the concept of "A man that is married to a doctor and has at least five children, all of whom are professors." This concept can be described with the following concept description:

$$\mathsf{Human} \sqcap \neg\mathsf{Female} \sqcap \exists\mathsf{married}.\mathsf{Doctor} \sqcap (\geq 5\,\mathsf{child}) \sqcap \forall\mathsf{child}.\mathsf{Professor}$$

This description employs the Boolean constructors *conjunction* ($\sqcap$), which is interpreted as set intersection, and *negation* ($\neg$), which is interpreted as set complement, as well as the *existential restriction* constructor ($\exists R.C$), the *value restriction* constructor ($\forall R.C$), and the *number restriction* constructor ($\geq n\,R$). An individual, say Bob, belongs to $\exists\mathsf{married}.\mathsf{Doctor}$ iff there exists an individual that is married to Bob (i.e., is related to Bob via the $\mathsf{married}$ role) and is a doctor (i.e., belongs to the concept Doctor). Similarly, Bob belongs to $(\geq 5\,\mathsf{child})$ iff he has at least five children, and he belongs to $\forall\mathsf{child}.\mathsf{Professor}$ iff all his children (i.e., all individuals related to Bob via the child role) are professors.

In addition to this description formalism, DLs are usually equipped with a terminological and an assertional formalism. In its simplest form, *terminological axioms* can be used to introduce names (abbreviations) for complex descriptions. For example, we could introduce the abbreviation $\mathsf{HappyMan}$ for the concept description from above. More expressive terminological formalisms allow the statement of constraints such as

$$\exists\mathsf{child}.\mathsf{Human} \sqsubseteq \mathsf{Human},$$

which says that only humans can have human children. The *assertional formalism* can be used to state properties of individuals. For example, the assertions

$$\mathsf{HappyMan}(\mathsf{BOB}), \quad \mathsf{child}(\mathsf{BOB}, \mathsf{MARY})$$

state that Bob belongs to the concept $\mathsf{HappyMan}$ and that Mary is one of his children.

Description Logic systems provide their users with various inference capabilities that deduce implicit knowledge from the explicitly represented knowledge. The *subsumption* algorithm determines subconcept-superconcept relationships: C is subsumed by D iff all instances of C are necessarily instances of D, i.e., the first description is always interpreted as a subset of the second description. For example, given the definition of $\mathsf{HappyMan}$ from above, $\mathsf{HappyMan}$ is subsumed by $\exists\mathsf{child}.\mathsf{Professor}$—since instances of $\mathsf{HappyMan}$ have at least five children, all of whom are professors, they also have a child that is a professor. The *instance* algorithm determines instance relationships: the individual i is an instance of the concept description C iff i is always interpreted as an element of C. For example, given the assertions from above and the definition of $\mathsf{HappyMan}$, MARY is an instance of $\mathsf{Professor}$. The *consistency* algorithm determines whether a knowledge base (consisting of a set of assertions and a set of terminological axioms) is non-contradictory. For example, if we add $\neg\mathsf{Professor}(\mathsf{MARY})$ to the two assertions from above, then the knowledge base containing these assertions together with the definition of $\mathsf{HappyMan}$ from above is inconsistent.

In order to ensure a reasonable and predictable behavior of a DL system, these inference problems should at least be decidable for the DL employed by the system, and preferably of low complexity. Consequently, the expressive power of the DL in question must be restricted in an appropriate way. If the imposed restrictions are too severe, however, then the important notions of the application domain can no longer be expressed. Investigating this trade-off between the expressivity of DLs and the complexity of their inference problems has been one of the most important issues in DL research. Roughly, the research related to this issue can be classified into the following four phases (see [?] for references).

Phase 1 (1980-1990) was mainly concerned with implementation of systems, such as Klone, K-Rep, Back, and Loom. These systems employed so-called *structural subsumption algorithms*, which first normalize the concept descriptions, and then recursively compare the syntactic structure of the normalized descriptions. These algorithms are usually very efficient (polynomial), but they have the disadvantage that they are complete only for very inexpressive DLs, i.e., for more expressive DLs they cannot detect all the existing subsumption/instance relationships. At the end of this phase, early formal investigations into the complexity of reasoning in DLs showed that most DLs do not have polynomial-time inference problems. As a reaction, the implementors of the Classic system (the first industrial-strength DL system) carefully restricted the expressive power of their DL.

Phase 2 (1990-1995) started with the introduction of a new algorithmic paradigm into DLs, so-called *tableau-based algorithms*. To decide the consistency of a knowledge base, a tableau-based algorithm tries to construct a model of it by breaking down the concepts in the knowledge base, thus inferring new constraints on the elements of this model. The algorithm either stops because all attempts to build a model failed with obvious contradictions, or it stops with a "canonical" model. Since subsumption and satisfiability can be reduced to consistency, a consistency algorithm can solve all inference problems mentioned above. In contrast to the structural algorithms of the first phase, tableau-based algorithms work on propositionally closed DLs (i.e., with all the Boolean operators) and are complete also for expressive DLs. The first systems employing such algorithms (Kris and Crack) demonstrated that optimized implementations of these algorithm lead to an acceptable behavior of the system, though their worst-case is no longer polynomial-time. This phase also saw a thorough analysis of the complexity of reasoning in various DLs. Another important observation was that DLs are very closely related to modal logics.

Phase 3 (1995-2000) is characterized by the development of inference procedures for very expressive DLs, either based on the tableau-approach or on a translation into modal logics. Highly optimized systems (FaCT, Race, and Dlp) showed that tableau-based algorithm for expressive DLs lead to a good practical behavior of the system even on (some) large knowledge bases. In this phase, the relationship to modal logics and to decidable fragments of first-order logic was also studied in more detail, and applications in databases (like schema reasoning, query optimization, and DB integration) were investigated.

We are now at the beginning of *Phase 4*, where industrial strength DL systems employing very expressive DLs and tableau-based algorithms are being developed, with applications like the Semantic Web or knowledge representation and integration in bio-informatics in mind.

Description Logics as Ontology Languages

As already mentioned above, high quality ontologies are crucial for the Semantic Web, and their construction, integration, and evolution greatly depends on the availability of a well-defined semantics and powerful reasoning tools. Since DLs provide for both, they should be ideal candidates for ontology languages. That much was already clear ten years ago, but at that time there was a fundamental mismatch between the expressive power and the efficiency of reasoning that DL systems provided, and the expressivity and the large knowledge bases that ontologists needed. Through the basic research in DLs of the last 10–15 years that we have summarized above, this gap between the needs of ontologist and the systems that DL researchers provide has finally become narrow enough to build stable bridges.

Regarding an ontology language for the Semantic Web, there is a joint US/EU initiative for a W3C ontology standard, for historical reasons called DAML+OIL [3, 2]. This language has a syntax based on RDF Schema (and thus is Web compatible), and it is based on common ontological primitives from Frame Languages (which supports human understandability). Its semantics can be defined via by a translation into the expressive DL $\mathcal{SHIQ}$ [**?**],[1] and the developers have tried to find a good compromise between expressiveness and the complexity of reasoning. Although reasoning in $\mathcal{SHIQ}$ is decidable, it has a rather high worst-case complexity (ExpTime). Nevertheless, there is a highly optimized $\mathcal{SHIQ}$ reasoner (FaCT) available, which behaves quite well in practice.

Let us point out some of the features of $\mathcal{SHIQ}$ that make reasoning hard. First, $\mathcal{SHIQ}$ provides number restrictions that are far more expressive than the ones introduced above (and employed be earlier DL systems). With the *qualified number restrictions* available in $\mathcal{SHIQ}$, as well as being able to say that a person has at most two children (without mentioning the properties of these children):

$$(\leq 2\,\mathsf{child}),$$

one can also specify that there is at most one son and at most one daughter:

$$(\leq 1\,\mathsf{child}.\neg\mathsf{Female}) \sqcap (\leq 1\,\mathsf{child}.\mathsf{Female})$$

Such concepts introduce additional non-determinism into reasoning since, given an individual for which the tableau-based algorithm has generated 3 children, one must *guess* which of them are female and which are not before one can detect the contradiction (see [**?**] for details).

[1]To be exact, the translation is into an extension of $\mathcal{SHIQ}$.

Second, $\mathcal{SHIQ}$ allows the formulation of complex terminological axioms like "humans have human parents":

$$\mathsf{Human} \sqsubseteq \exists \mathsf{parent}.\mathsf{Human}.$$

To test consistency of the assertion Human(BOB) under this constraint, a naive tableau-based algorithm would not terminate since it would generate an infinite chain of ancestors of Bob. To obtain a terminating procedure, one must check the computation for cycles by *blocking* the generation of role successors under certain conditions (see, e.g., [?]).

Third, $\mathcal{SHIQ}$ also allows for *inverse roles*; for example, in addition to child one can also use its inverse parent, with the connection between a role and its inverse being taken into account during reasoning. The combination of number restrictions, terminological axioms, and inverse roles is responsible for the fact that $\mathcal{SHIQ}$ no longer has the *finite model property*, i.e., there are subsumption relationships that do not hold, but for which the only counter-models are infinite models. To reason in such an expressive DL with a tableau-based algorithm, one must employ very sophisticated blocking techniques [?].

Conclusion

The emphasis in DL research on a formal, logic-based semantics and a thorough investigation of the basic reasoning problems, together with the availability of highly optimized systems for very expressive DLs, makes this family of knowledge representation formalisms an ideal starting point for defining ontology languages for the Semantic Web. The above mentioned reasoning services required to enable the construction, integration, and evolution of high quality ontologies are provided by state-of-the-are DL systems for very expressive languages.

To be used in practice, these languages will, however, also need DL-based tools that further support knowledge acquisition, maintenance, and other aspects of integration. First steps in this direction have already been taken. For example, OILEd is a tool that supports the development of OIL ontologies, and ICom is a tool that supports the integration of entity-relationship and UML diagrams. On a more fundamental level, so-called non-standard inferences that support building and maintaining knowledge bases (like computing least common subsumers, unification, and matching) are now an important topic of DL research [?]. All these efforts aim at supporting users that are not DL-experts in building and maintaining DL knowledge bases.

References

[1] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic Web. *Scientific American*, 284(5):34–43, May 2001.

[2] DAML language home page (`http://www.daml.org/language/`).

[3] I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In *Working Notes of the 2001 Int. Description Logics Workshop (DL-2001)*, pages 30–35. CEUR (`http://ceur-ws.org/`), volume 49, 2001.