Description Logics as Ontology Languages for the Semantic Web

Franz Baader[1], Ian Horrocks[2], and Ulrike Sattler[1]

[1] Theoretical Computer Science, RWTH Aachen, Germany
{baader,sattler}@cs.rwth-aachen.de
[2] Department of Computer Science, University of Manchester, UK
horrocks@cs.man.ac.uk

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 describe 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.

1 Introduction

The goal of this introduction is to sketch, on an informal level, what the Semantic Web is, why it needs ontologies, and where description logics come into play. Regarding the last point, we will first give a brief introduction to description logics, and then argue why they are well-suited as ontology languages. The remainder of this paper will then put some flesh on this skeleton by providing more technical details.

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 [15] 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. Such an annotation will be given in some standardized, expressive language (which, e.g., provides Boolean operators and some form of quantification) and make use of certain terms (like "Human", "Plant", etc.). To make sure that different agents have a common understanding of these terms, one needs *ontologies* in which these terms are described, and which thus establish a joint terminology between the agents. Basically, an ontology [44, 43] 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. Information on which concept is a specialization of another and which concepts are synonyms can be used in the design phase to test whether the concept definitions in the ontology have the intended consequences or not. Moreover, this information is also 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., when a Web page is 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 introduction to and history of description logics.

Description Logics

Description logics (DLs) [7, 24] 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.

In this introduction, we only illustrate some typical constructors by an example. Formal definitions are given in Section 2. 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:

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

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 ($\geq 5\,\mathsf{hasChild}$) iff he has at least five children, and he belongs to $\forall\mathsf{hasChild}.\mathsf{Professor}$ iff all his children (i.e., all individuals related to Bob via the $\mathsf{hasChild}$ 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

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

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

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

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 $\exists\mathsf{hasChild}.\mathsf{Professor}$—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.

Phase 1 (1980–1990) was mainly concerned with implementation of systems, such as KLONE, K-REP, BACK, and LOOM [19,61,70,60]. 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 [62]. 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 [18,63]. As a reaction, the implementors of the CLASSIC system (the first industrial-strength DL system) carefully restricted the expressive power of their DL [69,17].

Phase 2 (1990–1995) started with the introduction of a new algorithmic paradigm into DLs, so-called *tableau-based algorithms* [75,32,48]. They work on propositionally closed DLs (i.e., DLs with full Boolean operators) and are complete also for expressive DLs. 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 in propositionally closed DLs subsumption and satisfiability can be reduced to consistency, a consistency algorithm can solve all inference problems mentioned above. The first systems employing such algorithms (KRIS and CRACK) demonstrated that optimized implementations of these algorithms lead to an acceptable behavior of the system, though the worst-case complexity of the corresponding inference problem is no longer in polynomial time [6, 20]. This phase also saw a thorough analysis of the complexity of reasoning in various DLs [32–34]. Another important observation was that DLs are very closely related to modal logics [73].

Phase 3 (1995–2000) is characterized by the development of inference procedures for very expressive DLs, either based on the tableau-approach [56, 57] or on a translation into modal logics [29, 30, 28, 31]. Highly optimized systems (FaCT, Race, and Dlp [55, 45, 68]) 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 [29, 74] and to decidable fragments of first-order logic was also studied in more detail [16, 66, 42, 40, 41], and applications in databases (like schema reasoning, query optimization, and DB integration) were investigated [21, 22, 25, 26].

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 [35]. 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 [52, 27]. 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 by a translation into the expressive DL $\mathcal{SHIQ}$ [54],[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.

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

Let us point out some of the features of $\mathcal{SHIQ}$ that make this DL expressive enough to be used as an ontology language. Firstly, $\mathcal{SHIQ}$ provides number restrictions that are 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):

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

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

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

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

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

Thirdly, $\mathcal{SHIQ}$ also allows for *inverse roles*, *transitive roles*, and *subroles*. For example, in addition to $\mathsf{hasChild}$ one can also use its inverse $\mathsf{hasParent}$, one can specify that $\mathsf{hasAncestor}$ is transitive, and that $\mathsf{hasParent}$ is a subrole of $\mathsf{hasAncestor}$.

It has been argued in the DL and the ontology community that these features play a central role when describing properties of aggregated objects and when building ontologies [72, 76, 37]. The actual use of DLs providing these features as the underlying logical formalism of the web ontology languages OIL and DAML+OIL [36, 52] substantiates this claim [76].

2 The Expressive Description Logic $\mathcal{SHIQ}$

In contrast to most of the DLs considered in the literature, which concentrate on constructors for defining concepts, the DL $\mathcal{SHIQ}$ [53] also allows for rather expressive roles. Of course, these roles can then be used in the definition of concepts. We start with the definition of $\mathcal{SHIQ}$-roles, and then continue with the definition of $\mathcal{SHIQ}$-concepts.

Definition 1 (Syntax and semantics of $\mathcal{SHIQ}$-roles). *Let* $\mathbf{R}$ *be a set of* role names*, which is partitioned into a set* $\mathbf{R}_+$ *of transitive roles and a set* $\mathbf{R}_\mathsf{P}$ *of normal roles. The set of all* $\mathcal{SHIQ}$-roles *is* $\mathbf{R} \cup \{r^- \mid r \in \mathbf{R}\}$*, where* r^- *is called the* inverse *of the role* r*. A* role inclusion axiom *is of the form* $r \sqsubseteq s$*, where* r, s *are* $\mathcal{SHIQ}$*-roles. A* role hierarchy *is a finite set of role inclusion axioms.*

An interpretation $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *consists of a set* $\Delta^\mathcal{I}$*, called the* domain *of* $\mathcal{I}$*, and a function* $\cdot^\mathcal{I}$ *that maps every role to a subset of* $\Delta^\mathcal{I} \times \Delta^\mathcal{I}$ *such that, for all* $p \in \mathbf{R}$ *and* $r \in \mathbf{R}_+$*,*

$$\begin{aligned} \langle x, y\rangle \in p^\mathcal{I} \quad &\textit{iff} \quad \langle y, x\rangle \in (p^-)^\mathcal{I}, \\ \textit{if } \langle x, y\rangle \in r^\mathcal{I} \textit{ and } \langle y, z\rangle \in r^\mathcal{I} \quad &\textit{then} \quad \langle x, z\rangle \in r^\mathcal{I}. \end{aligned}$$

An interpretation $\mathcal{I}$ satisfies *a role hierarchy $\mathcal{R}$ iff $r^\mathcal{I} \subseteq s^\mathcal{I}$ for each $r \sqsubseteq s \in \mathcal{R}$; such an interpretation is called a* model *of $\mathcal{R}$.*

The unrestricted use of these roles in all of the concept constructors of $\mathcal{SHIQ}$ (to be defined below) would lead to an undecidable DL [53]. Therefore, we must first define an appropriate subset of all $\mathcal{SHIQ}$-roles. This requires some more notation.

1. The inverse relation on binary relations is symmetric, i.e., the inverse of r^- is again r. To avoid writing role expressions such as r^{--}, r^{---}, etc., we define a function Inv, which returns the inverse of a role:
$$\mathsf{Inv}(r) := \begin{cases} r^- & \text{if } r \text{ is a role name,} \\ s & \text{if } r = s^- \text{ for a role name } s. \end{cases}$$
2. Since set inclusion is transitive and an inclusion relation between two roles transfers to their inverses, a given role hierarchy $\mathcal{R}$ implies additional inclusion relationships. To account for this fact, we define $\overset{*}{\sqsubseteq}_\mathcal{R}$ as the reflexive-transitive closure of
$$\sqsubseteq_\mathcal{R} := \mathcal{R} \cup \{\mathsf{Inv}(r) \sqsubseteq \mathsf{Inv}(s) \mid r \sqsubseteq s \in \mathcal{R}\}.$$
We use $r \equiv_\mathcal{R} s$ as an abbreviation for $r \overset{*}{\sqsubseteq}_\mathcal{R} s$ and $s \overset{*}{\sqsubseteq}_\mathcal{R} r$. In this case, every model of $\mathcal{R}$ interprets these roles as the same binary relation.
3. Obviously, a binary relation is transitive iff its inverse is transitive. Thus, if $r \equiv_\mathcal{R} s$ and r or $\mathsf{Inv}(r)$ is transitive, then any model of $\mathcal{R}$ interprets s as a transitive binary relation. To account for such implied transitive roles, we define the following function Trans:
$$\mathsf{Trans}(s, \mathcal{R}) := \begin{cases} \text{true} & \text{if } r \in \mathbf{R}_+ \text{ or } \mathsf{Inv}(r) \in \mathbf{R}_+ \text{ for some } r \text{ with } r \equiv_\mathcal{R} s \\ \text{false} & \text{otherwise.} \end{cases}$$
4. A role r is called *simple* w.r.t. $\mathcal{R}$ iff $\mathsf{Trans}(s, \mathcal{R}) = \text{false}$ for all $s \overset{*}{\sqsubseteq}_\mathcal{R} r$.

Definition 2 (Syntax and semantics of $\mathcal{SHIQ}$-concepts). *Let N_C be a set of* concept names*. The set of $\mathcal{SHIQ}$-*concepts *is the smallest set such that*

1. *every concept name $A \in N_C$ is a $\mathcal{SHIQ}$-concept,*
2. *if C and D are $\mathcal{SHIQ}$-concepts and r is a $\mathcal{SHIQ}$-role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall r.C$, and $\exists r.C$ are $\mathcal{SHIQ}$-concepts,*
3. *if C is a $\mathcal{SHIQ}$-concept, r is a* simple *$\mathcal{SHIQ}$-role, and $n \in \mathbb{N}$, then $(\leqslant n\ r.C)$ and $(\geqslant n\ r.C)$ are $\mathcal{SHIQ}$-concepts.*

The interpretation function $\cdot^\mathcal{I}$ *of an interpretation* $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ *maps, additionally, every concept to a subset of* $\Delta^\mathcal{I}$ *such that*

$$(C \sqcap D)^\mathcal{I} = C^\mathcal{I} \cap D^\mathcal{I}, \qquad (C \sqcup D)^\mathcal{I} = C^\mathcal{I} \cup D^\mathcal{I}, \qquad \neg C^\mathcal{I} = \Delta^\mathcal{I} \setminus C^\mathcal{I},$$
$$(\exists r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \textit{There is some } y \in \Delta^\mathcal{I} \textit{ with } \langle x,y\rangle \in r^\mathcal{I} \textit{ and } y \in C^\mathcal{I}\},$$
$$(\forall r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \textit{For all } y \in \Delta^\mathcal{I}, \textit{ if } \langle x,y\rangle \in r^\mathcal{I}, \textit{then } y \in C^\mathcal{I}\},$$
$$(\leqslant n\ r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \sharp r^\mathcal{I}(x,C) \leqslant n\},$$
$$(\geqslant n\ r.C)^\mathcal{I} = \{x \in \Delta^\mathcal{I} \mid \sharp r^\mathcal{I}(x,C) \geqslant n\},$$

where $\sharp M$ *denotes the cardinality of the set* M*, and* $r^\mathcal{I}(x,C) := \{y \mid \langle x,y\rangle \in r^\mathcal{I} \text{ and } y \in C^\mathcal{I}\}$*. If* $x \in C^\mathcal{I}$*, then we say that* x *is an* instance *of* C *in* $\mathcal{I}$*, and if* $\langle x,y\rangle \in r^\mathcal{I}$*, then* y *is called an* r-successor *of* x *in* $\mathcal{I}$*.*

Concepts can be used to describe the relevant notions of an application domain. The terminology (TBox) introduces abbreviations (names) for complex concepts. In $\mathcal{SHIQ}$, the TBox allows one to state also more complex constraints.

Definition 3. *A* general concept inclusion *(GCI) is of the form* $C \sqsubseteq D$*, where* C, D *are* $\mathcal{SHIQ}$*-concepts. A finite set of GCIs is called a* TBox*. An interpretation* $\mathcal{I}$ *is a* model *of a TBox* $\mathcal{T}$ *iff it satisfies all GCIs in* $\mathcal{T}$*, i.e.,* $C^\mathcal{I} \subseteq D^\mathcal{I}$ *holds for each* $C \sqsubseteq D \in \mathcal{T}$*.*

A concept definition is of the form $A \equiv C$, where A is a concept name. It can be seen as an abbreviation for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$.

Inference problems are defined w.r.t. a TBox and a role hierarchy.

Definition 4. *The concept* C *is called* satisfiable with respect to the role hierarchy $\mathcal{R}$ and the TBox $\mathcal{T}$ *iff there is a model* $\mathcal{I}$ *of* $\mathcal{R}$ *and* $\mathcal{T}$ *with* $C^\mathcal{I} \neq \emptyset$*. Such an interpretation is called a* model of C *w.r.t.* $\mathcal{R}$ *and* $\mathcal{T}$*. The concept* D subsumes *the concept* C *w.r.t.* $\langle \mathcal{R}, \mathcal{T}\rangle$ *(written* $C \sqsubseteq_{\langle \mathcal{R}, \mathcal{T}\rangle} D$*) iff* $C^\mathcal{I} \subseteq D^\mathcal{I}$ *holds for all models* $\mathcal{I}$ *of* $\mathcal{R}$ *and* $\mathcal{T}$*. Two concepts* C, D *are* equivalent *w.r.t.* $\mathcal{R}$ *(written* $C \equiv_{\langle \mathcal{R}, \mathcal{T}\rangle} D$*) iff they subsume each other.*

By definition, equivalence can be reduced to subsumption. In addition, subsumption can be reduced to satisfiability since $C \sqsubseteq_{\langle \mathcal{R}, \mathcal{T}\rangle} D$ iff $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{R}$ and $\mathcal{T}$. Before sketching how to solve the satisfiability problem in $\mathcal{SHIQ}$, we try to give an intuition on how $\mathcal{SHIQ}$ can be used to define ontologies.

3 Describing Ontologies in $\mathcal{SHIQ}$

In general, an ontology can be formalised in a TBox as follows. Firstly, we restrict the possible worlds by introducing restrictions on the allowed interpretations. For example, to express that, in our world, we want to consider humans, which are either muggles or sorcerers, we can use the GCIs

$$\mathsf{Human} \sqsubseteq \mathsf{Muggle} \sqcup \mathsf{Sorcerer} \quad \text{and} \quad \mathsf{Muggle} \sqsubseteq \neg\mathsf{Sorcerer}.$$

Next, to express that humans have exactly two parents and that all parents and children of humans are human, we can use the following GCI:

$$\begin{aligned}\mathsf{Human} \sqsubseteq\ & \forall \mathsf{hasParent}.\mathsf{Human} \sqcap (\leqslant 2\ \mathsf{hasParent}.\top) \sqcap (\geqslant 2\ \mathsf{hasParent}.\top) \sqcap \\ & \forall \mathsf{hasParent}^{-}.\mathsf{Human},\end{aligned}$$

where $\top$ is an abbreviation for the top concept $A \sqcup \neg A$.

In addition, we consider the *transitive* role hasAncestor, and the role inclusion

$$\mathsf{hasParent} \sqsubseteq \mathsf{hasAncestor}.$$

The next GCI expresses that humans having an ancestor that is a sorcerer are themselves sorcerers:

$$\mathsf{Human} \sqcap \exists \mathsf{hasAncestor}.\mathsf{Sorcerer} \sqsubseteq \mathsf{Sorcerer}.$$

Secondly, we can define the relevant notions of our application domain using concept definitions. Recall that the concept definition $A \equiv C$ stands for the two GCIs $A \sqsubseteq C$ and $C \sqsubseteq A$. A concept name is called *defined* if it occurs on the left-hand side of a definition, and *primitive* otherwise.

We want our concept definitions to have definitional impact, i.e., the interpretation of the primitive concept and role names should uniquely determine the interpretation of the defined concept names. For this, the set of concept definitions together with the additional GCIs must satisfy three conditions:

1. There are no multiple definitions, i.e., each defined concept name must occur at most once as a left-hand side of a concept definition.
2. There are no cyclic definitions, i.e., no cyclic dependencies between the defined names in the set of concept definitions.[2]
3. The defined names do not occur in any of the additional GCIs.

In contrast to concept definitions, the GCIs in $\mathcal{SHIQ}$ may well have cyclic dependencies between concept names. An example are the above GCIs describing humans.

As a simple example of a set of concept definitions satisfying the restrictions from above, we define the concepts grandparent and parent:[3]

$$\begin{aligned}\mathsf{Parent} &\equiv \mathsf{Human} \sqcap \exists \mathsf{hasParent}^{-}.\top, \\ \mathsf{Grandparent} &\equiv \exists \mathsf{hasParent}^{-}.\mathsf{Parent},\end{aligned}$$

The TBox consisting of the above concept definitions and GCIs, together with the fact that hasAncestor is a transitive superrole of hasParent, implies the following subsumption relationship:

$$\mathsf{Grandparent} \sqcap \mathsf{Sorcerer} \sqsubseteq \exists \mathsf{hasParent}^{-}.\exists \mathsf{hasParent}^{-}.\mathsf{Sorcerer},$$

[2] In order to give cyclic definitions definitional impact, one would need to use fixpoint semantics for them [64, 2].

[3] In addition to the role hasParent, which relates children to their parents, we use the concept Parent, which describes all humans having children.

i.e., grandparents that are sorcerers have a grandchild that is a sorcerer. Though this conclusion may sound reasonable given the assumptions, it requires quite some reasoning to obtain it. In particular, one must use the fact that $\mathsf{hasAncestor}$ (and thus also $\mathsf{hasAncestor}^-$) is transitive, that $\mathsf{hasParent}^-$ is the inverse of $\mathsf{hasParent}$, and that we have a GCI that says that children of humans are again humans.

To sum up, a $\mathcal{SHIQ}$-TBox can, on the one hand, axiomatize the basic notions of an application domain (the primitive concepts) by GCIs, transitivity statements, and role inclusions, in the sense that these statements restrict the possible interpretations of the basic notions. On the other hand, more complex notions (the defined concepts) can be introduced by concept definitions. Given an interpretation of the basic notions, the concept definitions uniquely determine the interpretation of the defined notions.

The *taxonomy* of such a TBox is then given by the subsumption hierarchy of the defined concepts. It can be computed using a subsumption algorithm for $\mathcal{SHIQ}$ (see Section 5 below). The knowledge engineer can test whether the TBox captures her intuition by checking the satisfiability of the defined concepts (since it does not make sense to give a complex definition for the empty concept), and by checking whether their place in the taxonomy corresponds to their intuitive place. The expressive power of $\mathcal{SHIQ}$ together with the fact that one can "verify" the TBox in the sense mentioned above is the main reason for $\mathcal{SHIQ}$ being well-suited as an ontology language [72, 37, 76].

4 $\mathcal{SHIQ}$ and DAML+OIL

As already discussed, DAML+OIL is a semantic web ontology language whose semantics can be defined via a translation into an expressive DL. This is not a coincidence—it was a design goal. The mapping allows DAML+OIL to exploit formal results from DL research (e.g., regarding the decidability and complexity of key inference problems) and use implemented DL reasoners (e.g., FaCT [50] and Racer [46]) in order to provide reasoning services for DAML+OIL applications.

DAML+OIL uses a syntax that is based on RDF (the *Resource Description Framework*), and thus suitable for the Semantic Web. The underlying model for RDF is a labelled directed graph where nodes are either *resources* or *literals* (currently literals are just strings, but it is planed to extend the language to support type data values, e.g., "integer 5"). The graph is defined by a set of *triples*, statements of the form $\langle \mathsf{Subject}, \mathsf{Property}, \mathsf{Object} \rangle$, where $\mathsf{Subject}$ is a resource, $\mathsf{Property}$ is the edge label and Object is either a resource or a literal.

Everything describable by RDF is a resource; a resource may be named by a URI, but some resources (we will call them *anonymous resources*) may not be so named. A resource may be an entire Web page (identified by its URL), a part of a Web page (identified by its URL and an anchor), but also an object not accessible through the Web. A property is an attribute or relation used to describe a resource, and is also named by a URI. In practice, triples are written using a standard XML serialisation of RDF triples (see `http://www.w3.org/RDF/` for more details).

A DAML+OIL ontology can be seen to correspond to a DL TBox together with a role hierarchy, describing the domain in terms of *classes* (corresponding to concepts) and *properties* (corresponding to roles). An ontology consists of a set of *axioms* that assert, e.g., subsumption relationships between classes or properties. Asserting that an individual resource (a pair of resources) is an instance of a DAML+OIL class (property) is left to RDF, a task for which it is well suited.

As in a standard DLs, DAML+OIL classes may be names or expressions built up from simpler classes and properties using a variety of constructors. The set of constructors supported by DAML+OIL, along with the equivalent DL abstract syntax, is summarised in Figure 1.[4] The full XML serialisation of the RDF syntax is not shown as it is rather verbose, e.g., $\mathsf{Human} \sqcap \mathsf{Male}$ would be written as

```
<daml:Class>
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Human"/>
    <daml:Class rdf:about="#Male"/>
  </daml:intersectionOf>
</daml:Class>
```

while $(\geqslant 2\ \mathsf{hasChild.Lawyer})$ would be written as

```
<daml:Restriction daml:minCardinalityQ="2">
  <daml:onProperty rdf:resource="#hasChild"/>
  <daml:hasClassQ rdf:resource="#Lawyer"/>
</daml:Restriction>
```

Prefixes such as `daml:` specify XML namespaces for resources, while `rdf:parseType="daml:collection"` is a DAML+OIL extension to RDF that provides a "shorthand" notation for lisp style lists defined using triples with the properties first and rest (it can be eliminated, but with a consequent increase in verbosity). E.g., the first example above consists of the triples $\langle \mathsf{r_1}, \mathsf{daml:intersectionOf}, \mathsf{r_2} \rangle$, $\langle \mathsf{r_2}, \mathsf{daml:first}, \mathsf{Human} \rangle$, $\langle \mathsf{r_2}, \mathsf{rdfs:type}, \mathsf{Class} \rangle$, $\langle \mathsf{r_2}, \mathsf{daml:rest}, \mathsf{r_3} \rangle$, etc., where $\mathsf{r_i}$ is an anonymous resource, Human stands for a URI naming the resource "Human", and $\mathsf{daml:intersectionOf}$, $\mathsf{daml:first}$, $\mathsf{daml:rest}$ and $\mathsf{rdfs:type}$ stand for URIs naming the properties in question.

An important feature of DAML+OIL is that, besides "abstract" classes defined by the ontology, one can also use XML Schema *datatypes* (e.g., so called primitive datatypes such as string, decimal or float, as well as more complex derived datatypes such as integer sub-ranges) in `hasClass`, `hasValue`, and `cardinality`. E.g., the class Adult could be asserted to be equivalent to

[4] In fact, there are a few additional constructors provided as "syntactic sugar", but all are trivially reducible to the ones described in Figure 1.

Constructor	DL Syntax	Example
`intersectionOf`	$C_1 \sqcap \ldots \sqcap C_n$	$\mathsf{Human} \sqcap \mathsf{Male}$
`unionOf`	$C_1 \sqcup \ldots \sqcup C_n$	$\mathsf{Doctor} \sqcup \mathsf{Lawyer}$
`complementOf`	$\neg C$	$\neg\mathsf{Male}$
`oneOf`	$\{x_1 \ldots x_n\}$	$\{\mathsf{john}, \mathsf{mary}\}$
`toClass`	$\forall P.C$	$\forall\mathsf{hasChild.Doctor}$
`hasClass`	$\exists r.C$	$\exists\mathsf{hasChild.Lawyer}$
`hasValue`	$\exists r.\{x\}$	$\exists\mathsf{citizenOf}.\{\mathsf{USA}\}$
`minCardinalityQ`	$(\geqslant n\ r.C)$	$(\geqslant 2\ \mathsf{hasChild.Lawyer})$
`maxCardinalityQ`	$(\leqslant n\ r.C)$	$(\leqslant 1\ \mathsf{hasChild.Male})$
`inverseOf`	r^-	$\mathsf{hasChild}^-$

Fig. 1. DAML+OIL constructors

$\mathsf{Person} \sqcap \exists\mathsf{age.over17}$, where $\mathsf{over17}$ is an XML Schema datatype based on decimal, but with the added restriction that values must be at least 18. Using a combination of XML Schema and RDF this could be written as:

```
<xsd:simpleType name="over17">
  <xsd:restriction base="xsd:positiveInteger">
  <xsd:minInclusive value="18"/>
  </xsd:restriction>
</xsd:simpleType>

<daml:Class rdf:ID="Adult">
  <daml:intersectionOf rdf:parseType="daml:collection">
    <daml:Class rdf:about="#Person"/>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#age"/>
      <daml:hasClass rdf:resource="#over17"/>
    </daml:Restriction>
  </daml:intersectionOf>
</daml:Class>
```

As already mentioned, a DAML+OIL ontology consists of a set of axioms. Figure 2 summarises the axioms supported by DAML+OIL. These axioms make it possible to assert subsumption or equivalence with respect to classes or properties, the disjointness of classes, the equivalence or non-equivalence of individuals (resources), and various properties of properties. DAML+OIL also allows properties of properties (i.e., DL roles) to be asserted. In particular, it is possible to assert that a property is unique (i.e., functional), unambiguous (i.e., its inverse is functional) or transitive.

This shows that, except for individuals and datatypes, the constructors and axioms of DAML+OIL can be translated into $\mathcal{SHIQ}$. In fact, DAML+OIL is equivalent to the extension of $\mathcal{SHIQ}$ with nominals (i.e., individuals) and a

Axiom	DL Syntax	Example
`subClassOf`	$C_1 \sqsubseteq C_2$	Human $\sqsubseteq$ Animal $\sqcap$ Biped
`sameClassAs`	$C_1 \equiv C_2$	Man $\equiv$ Human $\sqcap$ Male
`subPropertyOf`	$P_1 \sqsubseteq P_2$	hasDaughter $\sqsubseteq$ hasChild
`samePropertyAs`	$P_1 \equiv P_2$	cost $\equiv$ price
`disjointWith`	$C_1 \sqsubseteq \neg C_2$	Male $\sqsubseteq \neg$Female
`sameIndividualAs`	$\{x_1\} \equiv \{x_2\}$	{President_Bush} $\equiv$ {G_W_Bush}
`differentIndividualFrom`	$\{x_1\} \sqsubseteq \neg\{x_2\}$	{john} $\sqsubseteq \neg${peter}
`transitiveProperty`	$P \in \mathbf{R}_+$	hasAncestor$^+ \in \mathbf{R}_+$
`uniqueProperty`	$\top \sqsubseteq (\leqslant 1\ P.\top)$	$\top \sqsubseteq (\leqslant 1$ hasMother.$\top)$
`unambiguousProperty`	$\top \sqsubseteq (\leqslant 1\ P^-.\top)$	$\top \sqsubseteq (\leqslant 1$ isMotherOf$^-.\top)$

Fig. 2. DAML+OIL axioms

simple form of so-called concrete domains [5]. This extension will be discussed in Section 6.

5 Reasoning in $\mathcal{SHIQ}$

Reasoning in $\mathcal{SHIQ}$ means deciding satisfiability and subsumption of $\mathcal{SHIQ}$-concepts w.r.t. TBoxes (i.e., sets of general concept inclusions) and role hierarchies. As shown in Section 2, subsumption can be reduced (in linear time) to satisfiability. In addition, since $\mathcal{SHIQ}$ allows for both subroles and transitive roles, TBoxes can be internalized, i.e., satisfiability w.r.t. a TBox and a role hierarchy can be reduced to satisfiability w.r.t. the empty TBox and a role hierarchy. In principle, this is achieved by introducing a (new) transitive superrole u of all roles occurring in the TBox $\mathcal{T}$ and the concept C_0 to be tested for satisfiability. Then we extend C_0 to the concept

$$\widehat{C}_0 := C_0 \sqcap \bigsqcap_{C \sqsubseteq D \in \mathcal{T}} (\neg C \sqcup D) \sqcap \forall u.(\neg C \sqcup D).$$

We can then show that $\widehat{C}_0$ is satisfiable w.r.t. the extended role hierarchy iff the original concept C_0 is satisfiable w.r.t. the TBox $\mathcal{T}$ and the original role hierarchy [1, 73, 3, 53].

Consequently, it is sufficient to design an algorithm that can decide satisfiability of $\mathcal{SHIQ}$-concepts w.r.t. role hierarchies and transitive roles. This problem is known to be ExpTime-complete [77]. In fact, ExpTime-hardness can be shown by an easy adaptation of the ExpTime-hardness proof for satisfiability in propositional dynamic logic [38]. Using automata-based techniques, Tobies [77] shows that satisfiability of $\mathcal{SHIQ}$-concepts w.r.t. role hierarchies is indeed decidable within exponential time.

In the remainder of this section, we sketch a tableau-based decision procedure for this problem. This procedure, which is described in more detail in [53], runs in worst case *nondeterministic* double exponential time. However, according to the current state of the art, this procedures is more practical than the ExpTime automata-based procedure in [77]. In fact, it is the basis for the highly optimised implementation of the DL system FaCT [51].

When started with a $\mathcal{SHIQ}$-concept C_0, a role hierarchy $\mathcal{R}$, and information on which roles are transitive, this algorithm tries to construct a model of C_0 w.r.t. $\mathcal{R}$. Since $\mathcal{SHIQ}$ has a so-called tree model property, we can assume that this model has the form of an infinite tree. If we want to obtain a decision procedure, we can only construct a finite tree representing the infinite one (if a (tree) model exists at all). This can be done such that the finite representation can be *unravelled* into an infinite tree model $\mathcal{I}$ of C_0 w.r.t. $\mathcal{R}$. In the finite tree representing this model, a node x corresponds to an individual $\pi(x) \in \Delta^{\mathcal{I}}$, and we label each node with the set of concepts $\mathcal{L}(x)$ that $\pi(x)$ is supposed to be an instance of. Similary, edges represent role-successor relationships, and an edge between x and y is labelled with the roles supposed to connect x and y. The algorithm either stops with a finite representation of a tree model, or with a *clash*, i.e., an obvious inconsistency, such as $\{C, \neg C\} \subseteq \mathcal{L}(x)$. It answers "$C_0$ is satisfiable w.r.t. $\mathcal{R}$" in the former case, and "C_0 is unsatisfiable w.r.t. $\mathcal{R}$" in the latter.

The algorithm is initialised with the tree consisting of a single node x labelled with $\mathcal{L}(x) = \{C_0\}$. Then it applies so-called *completion rules*, which break down the concepts in the node labels syntactically, thus inferring new constraints for the given node, and then extend the tree according to these constraints. For example, if $C_1 \sqcap C_2 \in \mathcal{L}(x)$, then the $\sqcap$-rule adds both C_1 and C_2 to $\mathcal{L}(x)$. The $\geq$-rule generates n new r-successor nodes $y_1, \ldots, y_n$ of x with $\mathcal{L}(y_i) = \{C\}$ if $(\geqslant n\ r.C) \in \mathcal{L}(x)$ and x does not yet have n distinct r-successors with C in their label. In addition, it asserts that these new successors must remain distinct (i.e., cannot be identified in later steps of the algorithm). Other rules are more complicated, and a complete description of this algorithm goes beyond the scope of this paper. However, we would like to point out two issues that make reasoning in $\mathcal{SHIQ}$ considerably harder than in less expressive DLs.

First, *qualified number restriction* are harder to handle than the unqualified ones used in most early DL systems. Let us illustrate this by an example. Assume that the algorithm has generated a node x with $(\leqslant 1\ \mathsf{hasChild}.\top) \in \mathcal{L}(x)$, and that this node has two $\mathsf{hasChild}$-successors y_1, y_2 (i.e., two edges labeled with $\mathsf{hasChild}$ leading to the nodes y_1, y_2). In order to satisfy the number restriction $(\leqslant 1\ \mathsf{hasChild}.\top)$ for x, the algorithm identifies node y_1 with node y_2 (unless these nodes were asserted to be distinct, in which case we have a clash). Now assume that we still have a node x with two $\mathsf{hasChild}$-successors y_1, y_2, but the label of x contains a qualified number restriction like $(\leqslant 2\ \mathsf{hasChild.Parent})$. The naive idea [78] would be to check the labels of y_1 and y_2 whether they contain Parent, and identify y_1 and y_2 only if both contain this concept. However, this is not correct since, in the model $\mathcal{I}$ constructed from the tree, $\pi(y_i)$ may well belong to $\mathsf{Parent}^{\mathcal{I}}$ even if this concept does not belong to the label of x. The first correct algorithm that can handle qualified number restrictions was proposed in [49]. The main idea is to introduce a so-called *choose-rule*. In our example, this rule would (nondeterministically) choose whether y_i is supposed to belong to Parent or $\neg\mathsf{Parent}$, and correspondingly extend its label. Together with the choose rule, the above naive identification rule is in fact correct.

Second, in the presence of *transitive roles*, guaranteeing termination of the algorithm is a non-trivial task [47, 71]. If $\forall r.C \in \mathcal{L}(x)$ for a transitive role r, then not only must we add C to the label of any r-successor y of x, but also $\forall r.C$. This ensures that, even over an "r-chain"

$$x \xrightarrow{r} y \xrightarrow{r} y_1 \xrightarrow{r} y_2 \xrightarrow{r} \ldots \xrightarrow{r} y_n$$

we get indeed $C \in \mathcal{L}(y_n)$. This is necessary since, in the model constructed from the tree generated by the algorithm, have

$$(\pi(x), \pi(y)),\ (\pi(y), \pi(y_1)),\ \ldots,\ (\pi(y_{n-1}), \pi(y_n)) \in r^{\mathcal{I}},$$

and thus the transitivity of $r^{\mathcal{I}}$ requires that also $(\pi(x), \pi(y_n)) \in r^{\mathcal{I}}$, and thus the value restriction on x applies to y_n as well. Propagating $\forall r.C$ over r-edges makes sure that this is taken care of. However, it also might lead to nontermination. For example, consider the concept $\exists r.A \sqcap \forall r.\exists r.A$ where r is a transitive role. It is easy to see that the algorithm then generates an infinite chain of nodes with label $\{A, \forall r.\exists r.A, \exists r.A\}$. To prevent this looping and ensure termination, we use a cycle-detection mechanism called *blocking*: if the labels of a node x and one of its ancestors coincide, we "block" the application of rules to x. The blocking condition must be formulated such that, whenever blocking occurs, we can "unravel" the blocked (finite) path into an infinite path in the model to be constructed. In description logics, blocking was first employed in [8] in the context of an algorithm that can handle GCIs, and was the improved on in [4, 23, 9]. In $\mathcal{SHIQ}$, the blocking condition is rather complicated since the combination of transitive and inverse roles r^- with number restrictions requires a rather advanced form of unravelling [53]. In fact, this combination of constructors is responsible for the fact that, unlike most DLs considered in the literature, $\mathcal{SHIQ}$ does not have the finite model property, i.e., there are satisfiable $\mathcal{SHIQ}$-concepts that are only satisfiable in infinite interpretations.

6 Extensions and variants of $\mathcal{SHIQ}$

As mentioned in Section 4, the ontology language DAML+OIL is a syntactic variant of $\mathcal{SHIQ}$ extended with nominals (i.e., concepts $\{x_1\}$ representing a singleton set consisting of one individual) and concrete datatypes (like a concept representing all integers between 4 and 17). In this section, we discuss the consequences of these extensions on the reasoning problems in $\mathcal{SHIQ}$.

Concrete datatypes, as available in DAML+OIL, are a very restricted form of so-called concrete domains [5]. For example, using the concrete domain of all nonnegative integers equipped with the $<$ predicate, a (functional) role age relating (abstract) individuals to their (concrete) age, and a (functional) subrole father of $\mathsf{hasParent}$, the following axiom states that children are younger than their fathers:

$$\mathsf{Animal} \sqsubseteq (\mathsf{age} < \mathsf{father} \circ \mathsf{age}).$$

Extending expressive DLs with concrete domains may easily lead to undecidability [10, 59]. However, DAML+OIL provides only a very limited form of concrete domains. In particular, the concrete domain must not allow for predicates of arity greater than 1 (like $<$ in our example), and the predicate restrictions must not contain role chains (like $\mathsf{father} \circ \mathsf{age}$ in our example). In [67], decidability of $\mathcal{SHIQ}$ extended with a slightly more general type of concrete domains is shown.

Concerning nominals, things become a bit more complicated. Firstly, it can be shown that $\mathcal{SHIQ}$ extended with nominals is a fragment of C2, the two-variable fragment of first order logic with counting quantifiers [39, 65, 77]. Thus, satisfiability and subsumption are decidable in NExpTime. This is optimal since the problem is also NExpTime-hard [77]. Roughly speaking, the combination of GCIs (or transitive roles and role hierarchies), inverse roles, and number restrictions with nominals is responsible for this leap in complexity (from ExpTime for $\mathcal{SHIQ}$ to NExpTime). To the best of our knowledge, no "practicable" decision procedure for $\mathcal{SHIQ}$ with nominals has been described until now. With "practicable" we mean an algorithm that can be implemented with reasonable effort and can be optimized such that it behaves well in practice (which is the case for the algorithm for $\mathcal{SHIQ}$ implemented in FaCT).

7 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 reasoning services required to support the construction, integration, and evolution of high quality ontologies are provided by state-of-the-art 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 (i.e., building ontologies), maintenance (i.e., evolution of ontologies), and integration and inter-operation of ontologies. First steps in this direction have already been taken. For example, OilEd [14] is a tool that supports the development of OIL[5] and DAML+OIL ontologies, and ICom is a tool that supports the design and 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 [12, 13, 11, 58]. All these efforts aim at supporting users that are not DL-experts in building and maintaining DL knowledge bases.

[5] OIL is a fragment of DAML+OIL.

References

1. F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, 1991.
2. F. Baader. Using automata theory for characterizing the semantics of terminological cycles. *Annals of Mathematics and Artificial Intelligence*, 18(2–4):175–219, 1996.
3. F. Baader, H.-J. Bürckert, B. Nebel, W. Nutt, and G. Smolka. On the expressivity of feature logics with negation, functional uncertainty, and sort equations. *Journal of Logic, Language and Information*, 2:1–18, 1993.
4. F. Baader, H.-J. Bürkert, B. Hollunder, W. Nutt, and J. H. Siekmann. Concept logics. In John W. Lloyd, editor, *Computational Logics, Symposium Proceedings*, pages 177–201. Springer-Verlag, 1990.
5. F. Baader and P. Hanschke. A schema for integrating concrete domains into concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 452–457, Sydney, 1991.
6. F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithm. In *Proc. of the Workshop on Processing Declarative Knowledge, PDK-91*, volume 567 of *Lecture Notes In Artificial Intelligence*, pages 67–86. Springer-Verlag, 1991.
7. F. Baader and U. Sattler. An overview of tableau algorithms for description logics. *Studia Logica*, 2001. To appear. An abridged version appeared in *Tableaux 2000*, volume 1847 of LNAI, 2000. Springer-Verlag.
8. F. Baader. Augmenting concept languages by transitive closure of roles: An alternative to terminological cycles. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, 1991.
9. F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence Journal*, 88(1–2):195–213, 1996.
10. F. Baader and P. Hanschke. Extensions of concept languages for a mechanical engineering application. In *Proc. of the 16th German AI-Conference, GWAI-92*, volume 671 of *Lecture Notes in Computer Science*, pages 132–143, Bonn, Germany, 1992. Springer-Verlag.
11. F. Baader, R. Küsters, A. Borgida, and D. L. McGuinness. Matching in description logics. *Journal of Logic and Computation*, 9(3):411–447, 1999.
12. F. Baader, R. Küsters, and R. Molitor. Computing least common subsumers in description logics with existential restrictions. In *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, pages 96–101, 1999.
13. F. Baader and P. Narendran. Unification of concepts terms in description logics. *J. of Symbolic Computation*, 31(3):277–305, 2001.
14. S. Bechhofer, I. Horrocks, C. Goble, and R. Stevens. OilEd: a reason-able ontology editor for the semantic web. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 1–9. CEUR (`http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/`), 2001.
15. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic Web. *Scientific American*, 284(5):34–43, 2001.
16. A. Borgida. On the relative expressive power of Description Logics and Predicate Calculus. To appear in *Artificial Intelligence*, 1996.
17. R. J. Brachman. “reducing” CLASSIC to practice: Knowledge representation meets reality. In *Proc. of the 3rd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-92)*, pages 247–258. Morgan Kaufmann, Los Altos, 1992.

18. R. J. Brachman and H. J. Levesque. The tractability of subsumption in frame-based description languages. In *Proc. of the 4th Nat. Conf. on Artificial Intelligence (AAAI-84)*, pages 34–37, 1984.
19. R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.
20. P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: Preliminary report. In *Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.
21. M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf. Terminological systems revisited: Terminology = schema + views. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pages 199–204, Seattle (USA), 1994.
22. M. Buchheit, F. M. Donini, W. Nutt, and A. Schaerf. A refined architecture for terminological systems: Terminology = schema + views. *Artificial Intelligence Journal*, 99(2):209–260, 1998.
23. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.
24. D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi. Reasoning in expressive description logics. In A. Robinson and A. Voronkov, editors, *Handbook of Automated Reasoning*. Elsevier Science Publishers (North-Holland), Amsterdam, 1999.
25. D. Calvanese, G. De Giacomo, and M. Lenzerini. On the decidability of query containment under constraints. In *Proc. of the Seventeenth ACM SIGACT SIGMOD Sym. on Principles of Database Systems (PODS-98)*, pages 149–158, 1998.
26. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Description logic framework for information integration. In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 2–13, 1998.
27. DAML language home page (`http://www.daml.org/language/`).
28. G. De Giacomo. *Decidability of Class-Based Knowledge Representation Formalisms*. PhD thesis, Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", 1995.
29. G. De Giacomo and M. Lenzerini. Boosting the correspondence between description logics and propositional dynamic logics. In *Proc. of the 12th Nat. Conf. on Artificial Intelligence (AAAI-94)*, pages 205–212. AAAI Press/The MIT Press, 1994.
30. G. De Giacomo and M. Lenzerini. Concept language with number restrictions and fixpoints, and its relationship with μ-calculus. In *Proc. of the 11th European Conf. on Artificial Intelligence (ECAI-94)*, pages 411–415, 1994.
31. G. De Giacomo and M. Lenzerini. TBox and ABox reasoning in expressive description logics. In Luigia C. Aiello, John Doyle, and Stuart C. Shapiro, editors, *Proc. of the 5th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-96)*, pages 316–327. Morgan Kaufmann, Los Altos, 1996.
32. F. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, Boston, MA, USA, 1991.
33. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. Tractable concept languages. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 458–463, Sydney, 1991.
34. F. M. Donini, B. Hollunder, M. Lenzerini, A. M. Spaccamela, D. Nardi, and W. Nutt. The complexity of existential quantification in concept languages. *Artificial Intelligence Journal*, 2–3:309–327, 1992.

35. J. Doyle and R. S. Patil. Two theses of knowledge representation: Language restrictions, taxonomic classification, and the utility of representation services. *Artificial Intelligence Journal*, 48:261–297, 1991.
36. D. Fensel, F. van Harmelen, I. Horrocks, D. McGuinness, and P. F. Patel-Schneider. OIL: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, 16(2):38–45, 2001.
37. D. Fensel, F. van Harmelen, M. Klein, H. Akkermans, J. Broekstra, C. Fluit, J. van der Meer, H.-P. Schnurr, R. Studer, J. Hughes, U. Krohn, J. Davies, R. Engels, B. Bremdal, F. Ygge, T. Lau, B. Novotny, U. Reimer, and I. Horrocks. On-To-Knowledge: Ontology-based tools for knowledge management. In *Proceedings of the eBusiness and eWork 2000 (eBeW'00) Conference*, 2000.
38. M. J. Fischer and R. E. Ladner. Propositional dynamic logic of regular programs. *Journal of Computer and System Science*, 18:194–211, 1979.
39. E. Grädel, M. Otto, and E. Rosen. Two-variable logic with counting is decidable. In *Proc. of the 12th Ann. IEEE Symp. on Logic in Computer Science (LICS-97)*, 1997. Available via `http://speedy.informatik.rwth-aachen.de/WWW/papers.html`.
40. E. Grädel. Guarded fragments of first-order logic: A perspective for new description logics? In *Proc. of the 1998 Description Logic Workshop (DL'98)*. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-11/, 1998.
41. E. Grädel. On the restraining power of guards. *Journal of Symbolic Logic*, 64:1719–1742, 1999.
42. E. Grädel, Phokion G. Kolaitis, and Moshe Y. Vardi. On the decision problem for two-variable first-order logic. *Bulletin of Symbolic Logic*, 3(1):53–69, 1997.
43. T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.
44. N. Guarino. Formal ontology, conceptual analysis and knowledge representation. *Int. Journal of Human-Computer Studies*, 43(5/6):625–640, 1995.
45. V. Haarslev and R. Möller. RACE system description. In P. Lambrix, A. Borgida, M. Lenzerini, R. Möller, and P. Patel-Schneider, editors, *Proceedings of the International Workshop on Description Logics*, Linköping, Sweden, 1999. CEUR.
46. V. Haarslev and R. Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, volume 2083 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 2001.
47. J. Y. Halpern and Y. Moses. A guide to completeness and complexity for modal logic of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
48. B. Hollunder, W. Nutt, and M. Schmidt-Schauss. Subsumption algorithms for concept description languages. In *ECAI-90*, Pitman Publishing, London, 1990.
49. B. Hollunder and F. Baader. Qualifying number restrictions in concept languages. In *Proc. of the 2nd Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-91)*, pages 335–346, 1991.
50. I. Horrocks. The FaCT system. In Harrie de Swart, editor, *Proc. of the Int. Conf. on Automated Reasoning with Analytic Tableaux and Related Methods (TABLEAUX-98)*, volume 1397 of *Lecture Notes In Artificial Intelligence*, pages 307–312. Springer-Verlag, 1998.
51. I. Horrocks. Using an Expressive Description Logic: FaCT or Fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, 1998.

52. I. Horrocks and P. Patel-Schneider. The generation of DAML+OIL. In *Proc. of the 2001 Description Logic Workshop (DL 2001)*, pages 30–35. CEUR (`http://ceur-ws.org/`), volume 49, 2001.
53. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In H. Ganzinger, D. McAllester, and A. Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes In Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
54. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic shiq. In D. MacAllester, editor, *Proc. of the 17th Conf. on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Computer Science, Germany, 2000. Springer-Verlag.
55. I. Horrocks. Using an expressive description logic: FaCT or fiction? In *Proc. of the 6th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR-98)*, pages 636–647, 1998.
56. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
57. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In Harald Ganzinger, David McAllester, and Andrei Voronkov, editors, *Proc. of the 6th Int. Conf. on Logic for Programming and Automated Reasoning (LPAR'99)*, number 1705 in Lecture Notes In Artificial Intelligence, pages 161–180. Springer-Verlag, 1999.
58. R. Küsters. *Non-Standard Inferences in Description Logics*, volume 2100 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 2001.
59. C. Lutz. NExpTime-complete description logics with concrete domains. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR-01)*, number 2083 in Lecture Notes In Artificial Intelligence, pages 45–60. Springer-Verlag, 2001.
60. R. MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.
61. E. Mays, R. Dionne, and R. Weida. K-REP system overview. *SIGART Bulletin*, 2(3), 1991.
62. B. Nebel. *Reasoning and Revision in Hybrid Representation Systems*. Lecture Notes In Artificial Intelligence. Springer-Verlag, 1990.
63. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence Journal*, 43:235–249, 1990.
64. B. Nebel. Terminological cycles: Semantics and computational properties. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 331–361. Morgan Kaufmann, Los Altos, 1991.
65. L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In *Proc. of the 12th Ann. IEEE Symp. on Logic in Computer Science (LICS-97)*, 1997.
66. L. Pacholski, W. Szwast, and L. Tendera. Complexity of two-variable logic with counting. In *Proc. of the 12th Ann. IEEE Symp. on Logic in Computer Science (LICS-97)*, pages 318–327. IEEE Computer Society Press, 1997.
67. J. Z. Pan. Web ontology reasoning in the SHOQ(D) description logic. In *Proceedings of the Workshop on Methods for Modalities 2001 (M4M-2001)*, Amsterdam, 2001. ILLC.

68. P. F. Patel-Schneider. DLP. In *Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13. CEUR Electronic Workshop Proceedings, http://ceur-ws.org/Vol-22/, 1999.
69. P. F. Patel-Schneider, D. L. McGuiness, R. J. Brachman, L. A. Resnick, and A. Borgida. The CLASSIC knowledge representation system: Guiding principles and implementation rational. *SIGART Bulletin*, 2(3):108–113, 1991.
70. C. Peltason. The BACK system — an overview. *SIGART Bulletin*, 2(3):114–119, 1991.
71. U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, volume 1137 of *Lecture Notes In Artificial Intelligence*. Springer-Verlag, 1996.
72. U. Sattler. Description logics for the representation of aggregated objects. In W.Horn, editor, *Proceedings of the 14th European Conference on Artificial Intelligence*. IOS Press, Amsterdam, 2000.
73. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of the 12th Int. Joint Conf. on Artificial Intelligence (IJCAI-91)*, pages 466–471, Sydney, 1991.
74. K. Schild. *Querying Knowledge and Data Bases by a Universal Description Logic with Recursion*. PhD thesis, Universität des Saarlandes, Germany, 1995.
75. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence Journal*, 48(1):1–26, 1991.
76. R. Stevens, I. Horrocks, C. Goble, and S. Bechhofer. Building a reason-able bioinformatics ontology using OIL. In *Proceedings of the IJCAI-2001 Workshop on Ontologies and Information Sharing*, pages 81–90, 2001.
77. S. Tobies. *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*. PhD thesis, RWTH Aachen, 2001. electronically available at http://www.bth.rwth-aachen.de/ediss/ediss.html.
78. W. van der Hoek and M. De Rijke. Counting objects. *Journal of Logic and Computation*, 5(3):325–345, 1995.